Preliminary Design for a Spatial Data Management System

(13)

Technical Report
CCA-78-09
June 30, 1978

Christopher F. Herot
Jim Schmolze
Richard Carling
Jerry Farrell

Computer Corporation of America
575 Technology Square
Cambridge, Massachusetts 02139
617-491-3670

(6)

Preliminary Design for a Spatial Data Management System

(10)

Christopher F. Herot
Jim Schmolze
Richard Carling
Jerry Farrell

(9)

Technical Report.

(14) CCA-78-09

(11) 30 June 30, 1978

(12) 153p.

Computer Corporation of America
575 Technology Square
Cambridge, Massachusetts 02139

This research was supported by the Advanced Research Projects Agency of the Department of Defense under Contract No. (15) MDA-903-78-C-0122, ARPA Order — 3487. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U.S. Government.

78 07 14 021

387 285

LB

Table of Contents

1. INTRODUCTION

Spatial Data Management is a technique for organizing and retrieving information by positioning it in a Graphical Data Space. In contrast to conventional database management systems (DBMSs) which require that information be stored and retrieved by specifying attributes as numbers or strings of text, Spatial Data Management allows a user to employ the spatial location and visual appearance of information in order to find it. The underlying concept is that spatial location is, for many purposes, easier to remember and work with than the keywords of an ordinary DBMS. By allowing a datum to be stored in proximity to related pieces of information, a Spatial Data Management System (SDMS) frees the user of the need to know the exact name or location of the information that he seeks. Instead, he can locate it by "browsing" until he finds something that he can identify visually. Spatial management of data can be combined with conventional DBMS techniques to yield a system which provides both means of access to information. Such a system is envisaged to have widespread application in the management of any complex activity where efficient and quick access to large and complex databases is required, such as command and control or intelligence analysis.

This document describes the functional capabilities of a prototype SDMS and sets forth the preliminary design of the implementation of these capabilities. The system capitalizes on pioneering work done at the Architecture Machine Group at MIT [NEGROPONTE & FIELDS], [BOLT a], [DONELSON], extending that work with the goal of realizing a system which performs a useful function in real DoD applications.

This chapter introduces the concept of an SDMS in general, and it details the relationship between the MIT and CCA efforts. The remainder of the report describes the CCA system in detail, including the particular means by which the user can store and access information in the Graphical Data Space and its related symbolic database.

Chapter 2 gives a detailed example of SDMS usage. The reader may wish to return to it after reading the remainder of the report. It is included at the beginning in order to impart a flavor for the type of interaction which SDMS affords.

Chapter 3 describes the operations a user can perform in order to retrieve data using the graphical capabilities of SDMS.

Chapter 4 describes the navigational aids which can be used to help a user find his way through the SDMS.

Chapter 5 introduces INGRES, the symbolic database management system, and describes how data which is stored in it may be represented graphically and organized spatially.

Chapter 6 introduces the mechanism through which symbolic and graphical data are associated.

Chapter 7 describes Icon Class Definition Language, the facility which allows the database administrator to define spatial and graphical representations of symbolic data.

Chapter 8 presents SQUEL, the graphical and symbolic query language of SDMS.

Chapter 9 summarizes all of the commands which are provided to interact with SDMS, for both users and the database administrator.

Chapter 10 briefly discusses the implementation of the system, a topic which will be covered in more detail in a later report.

Appendix A shows in detail the Icon Definition Language which generates the I-spaces of Chapter 2.

Finally, a glossary is presented as Appendix B.

1.1 Concepts

The Graphical Data Space (GDS) of an SDMS is partitioned into Information Spaces, or I-Spaces. An I-Space is the basic mechanism for storing information in an SDMS. The user may place data in an I-Space and search for it in much the same way that he would use the top of his desk. An I-Space appears as a flat, two-dimensional plane of colored images, presented on a television monitor driven by a raster-scan frame-buffer display.*

As an I-Space is typically larger than what can be seen at any one time on a television monitor, the user is given the capability of moving over the plane of data through the use of joy sticks, moving about as if he were piloting a helicopter. One joy stick controls motion parallel to the surface of the data causing it to scroll, that is, new information appears at one margin as the contents of the

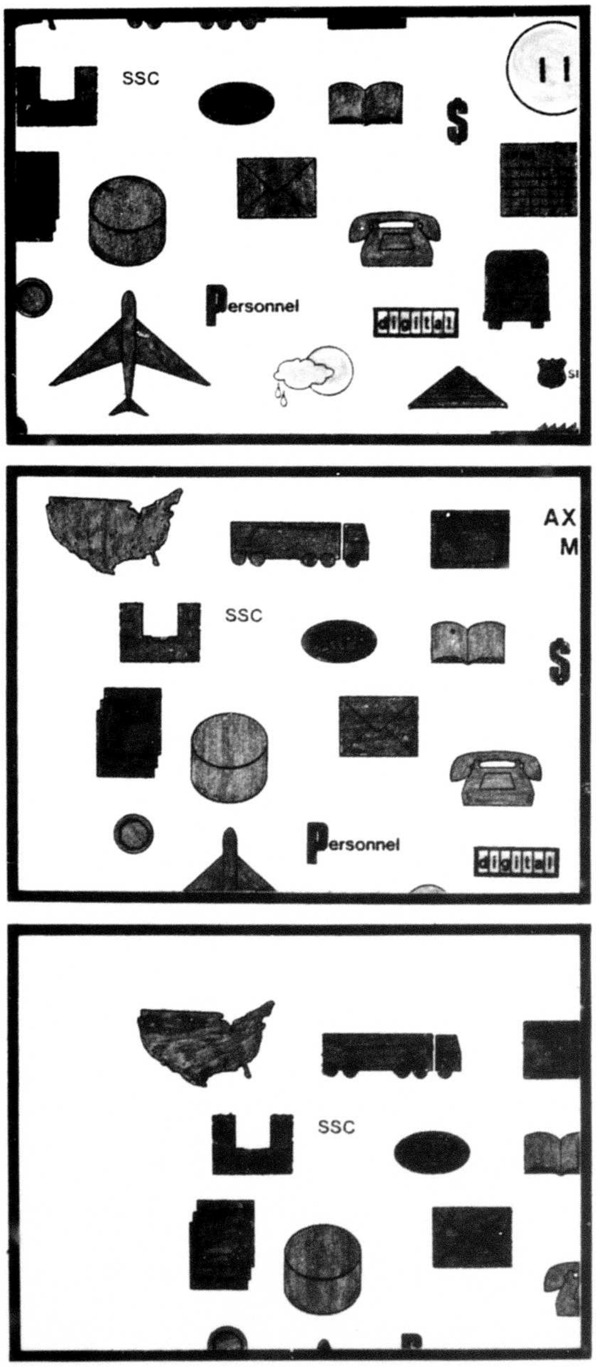
---

* Such displays employ a memory with one cell for each picture element (pixel). The value stored in a particular cell determines the color displayed at the associated pixel position. The memory or "frame buffer" is read out once per video frame, typically 30 times per second. Writing the image is normally a much slower process.

display disappear off the opposite margin (see Figure 1.1). The other joy stick controls motion perpendicular to the plane, allowing the user to descend for a closer look or back off for a more encompassing view of the data.

Information may reside in the I-Space itself, in the form of photographs, diagrams, strings of text, sound, or animation. It may also be stored in a symbolic DBMS as tuples in a relation or records in a dataset. While certain types of data may be more suited to either the graphical or symbolic forms, significant advantages accrue from storing the data redundantly in both. In this way, a user who is uncertain of the exact name of a particular attribute may search among data known to be related to the sought data. He may also "browse" in the hope of uncovering something which he might never find if he were required to specify it precisely. On the other hand, the user may avoid the time-consuming perusal of a large set of photographs by issuing a query to a symbolic database system specifying the nature of the photograph he seeks.

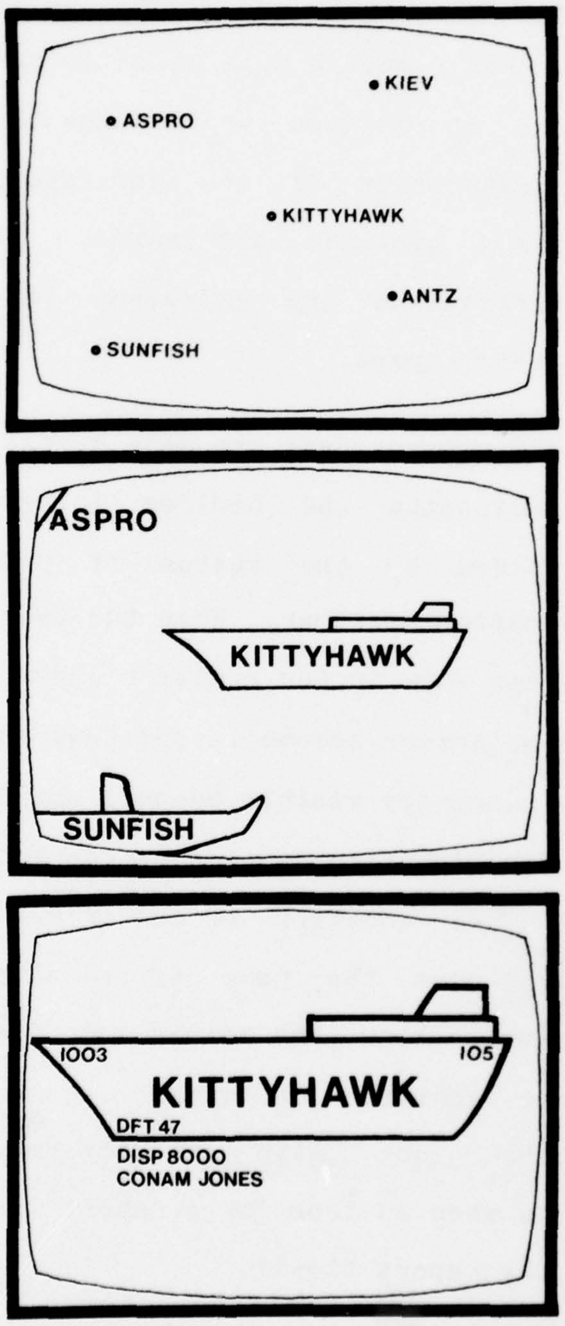
When the user first views an I-Space, he usually does not see the data itself. Rather, he sees a set of <u>icons</u>, images chosen to be representative of the data but easier to recognize and occupying less space. For example, data about a ship could be indicated by a picture of a ship, Scrolling about an I-Space of icons Figure 1.1



financial information by a dollar sign, and geographical information by a picture of a globe (as in Figure 1.1). These icons are constructed as information is entered into the database. For symbolic data which is part of a DBMS, procedures can be defined which make the size, shape, color, and orientation of the corresponding icon a function of its symbolic attributes. The language in which these definitions are expressed is described in Chapter 7 of this report.

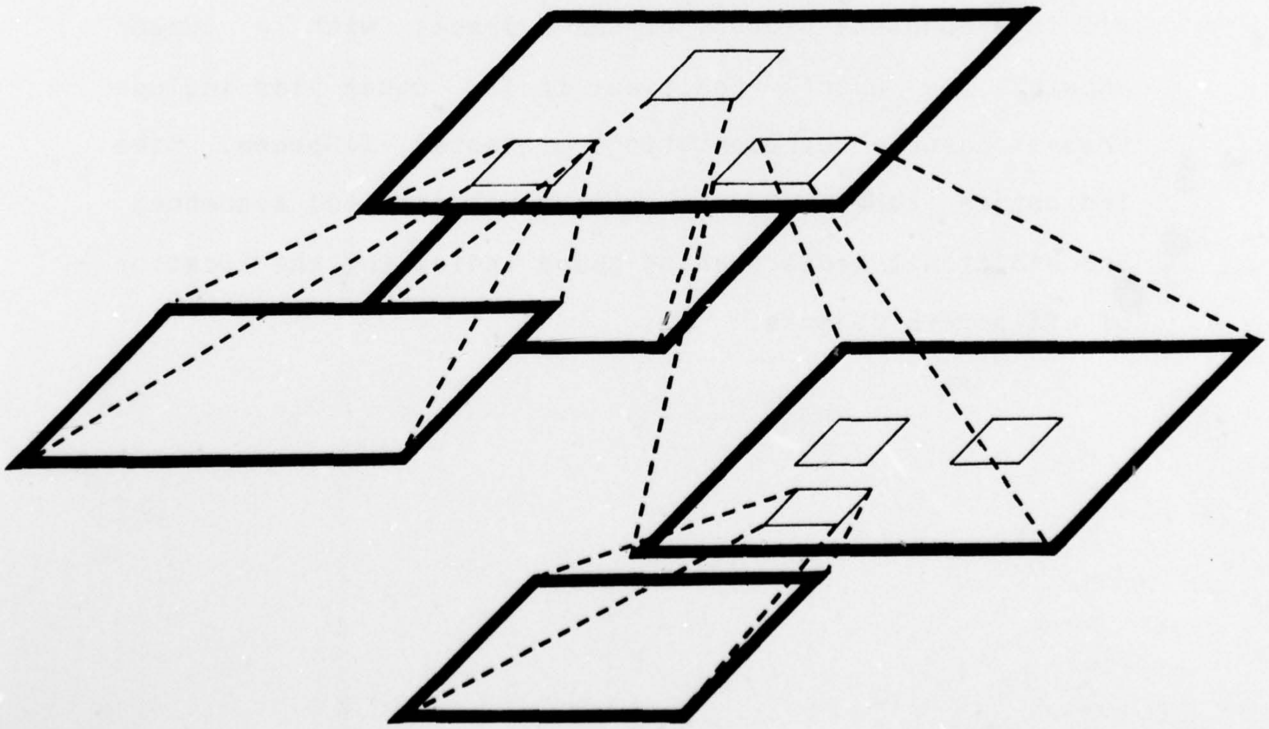
As the user maneuvers over the data surface, he can look for icons indicating the desired information. This process is aided by the custom of placing icons for related information together. When the user locates such an icon, he can zoom in for a closer look. As the chosen icon and its neighbors become larger (and the portion of the I-Space currently visible becomes smaller) the system gives him a more detailed version of the icon. Sometimes this entails the addition of one or more new pieces of information, as when the name of a ship's commanding officer appears below the ship's icon (see Figure 1.2). Sometimes the addition of detail is accomplished by replacing the icon with the information which it represents, as when an icon for a report is replaced by the text of the report itself.

Addition of Detail to an Icon Figure 1.2



To organize information in ways which are more complex than a two-dimensional layout, users may nest I-Spaces inside each other through a mechanism called a port. A port is an area of the I-Space, usually associated with an icon, which allows the user to leave that I-Space. When a port is activated by zooming into it, the user enters a new environment. This new environment can, in turn, be a new I-Space.

Nested I-Spaces Figure 1.3



Ports can also be used to activate any arbitrary process of the host operating system. Such a process can make use of the display and the database to perform its functions. This facility allows the user to enter a perusal space in which he can read and edit text, watch animated sequences, listen to and record messages, etc.

In order to help the user find his way through the SDMS, a variety of navigational aids are provided. The primary aid is a complete picture of the I-Space, with a cursor showing the user's position in it. Other aids include tree-structured representations of nested I-Spaces, time indicators showing progress through animated sequences, and additional media such as sound indicating the location of off-screen objects.

1.2 Background

1.2.1 The MIT System

The Architecture Machine Group at MIT is an interdisciplinary laboratory working on problems of man-machine interaction. The work there has been in the exploration of new modes of multi-media presentation and interaction, especially raster-scan computer graphics. It was in this environment that work began on the first system for the spatial management of data. The MIT SDMS effort, now a year old, has produced an initial working system [BOLT a] and is in the process of constructing a more advanced version [DONELSON].

The MIT researchers solved the problem of simulating motion through space by constructing their own raster-scan display. This device, based on an Interdata model 85, provides 256K bytes of MOS memory which is shared between the processor and the display. It is one of the first frame buffers in existence and the first to offer the hardware scrolling and zooming facilities which make possible the appearance of motion over the data surface.

The SDMS which MIT implemented presents to the user a world of nested I-Spaces connected by ports. The size of each I-Space is constrained by the amount of information which can be wholly contained in the 85's memory at one time. The I-Space is populated with icons, usually in one-to-one correspondence with ports. Zooming in on such a port causes the user to pop through to its associated I-Space. The change is not continuous, but results in sudden transitions from an icon to its associated data. The new system currently under construction will provide for gradually adding detail by allowing the creator of the database to place similar, but more detailed, icons on the I-Spaces under the ports. However, once the user passes through a port, he must return through that port. It is not possible to zoom in on one icon and then move over to the detailed version of another one.

The MIT SDMS can only handle data in graphical form. It has no companion database management system for storage and retrieval of symbolic information. The range of graphical data types is quite broad, however. It currently includes digitized photographs and video with synchronized sound. In fact, the entire user environment is organized towards new modes and media of interaction. The user sits in a "media room" in front of a 6 ft. x 9 ft. rear-projected screen. The arms of his chair are outfitted with joy sticks and small touch-sensitive data tablets. A larger stylus-driven tablet can be placed in his lap and two small color monitors are placed on a pedestal slightly to one side. The room itself is fitted with a stereo sound system under computer control. The entire configuration is supported by an array of four large minicomputers, as well as an assortment of video equipment such as NTSC encoders, vidicons, disks, mixers and faders.

This unique environment proved ideal for the demonstration of the potential of spatial data management. It demonstrated the utility of a two-dimensional, color, raster-scan graphical data space for storing and retrieving information. Most users found the system easy to learn and natural to use. While the basic concept changed little during the first year of research, many important discoveries were made, such as the importance of "navigational aids" for finding one's way about the graphical data space [BOLT b]. The CCA prototype described in the remainder of this report adheres rather closely to the concept embodied in the MIT system, but with certain key changes necessitated by the transition from university laboratory to operational environment.

1.2.2 The CCA System

The CCA SDMS effort is targeted towards the production of an operational prototype which can be used to evaluate the utility of the Spatial Data Management concept in a more operational setting than that possible at MIT. The system must meet the following new criteria:

1. It must have a good interface to a symbolic database management system (DBMS). This permits the user to employ symbolic queries to find spatial data and spatial search to find symbolic data. In addition, it simplifies the input of large amounts of data, reducing the need for "hand crafting" the associated graphical data space.

2. The system must be robust, well documented, easily maintainable and lend itself to replication. All of these factors mitigate heavily in favor of commercially available hardware.

3. It must be designed with a certain amount of hardware independence, so as to accommodate new capabilities which are appearing rapidly, such as 1000-line television.

4. It must lend itself to replication at a modest cost, certainly less than the cost of the four large size minicomputers used at MIT.

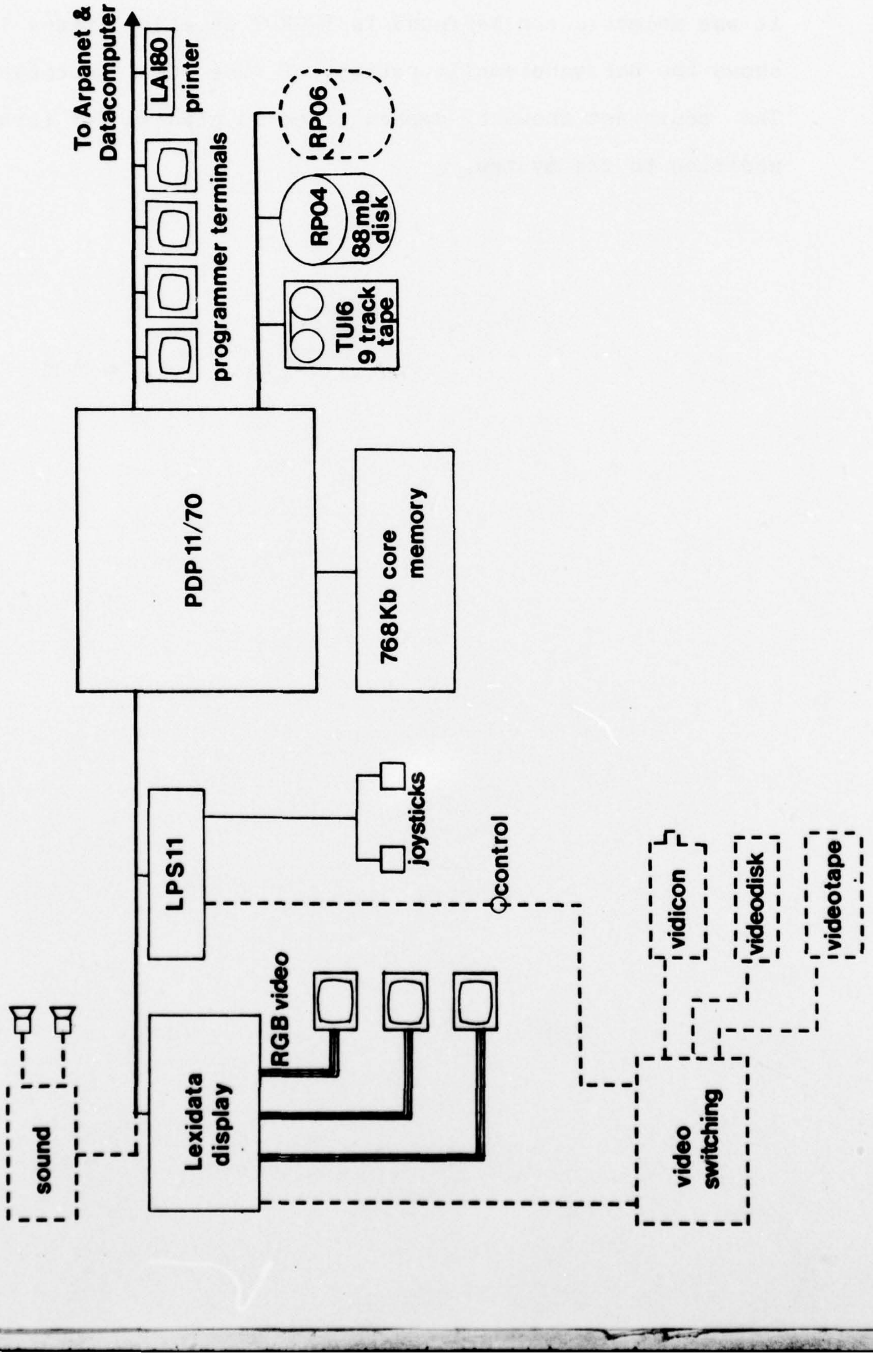
The addition of a symbolic database involves the largest departure from the MIT system. By providing such a facility, the prototype not only offers a capability not present in the MIT system, but something which is not in any existing database system. The ability to make graphical searches of a symbolic database allows a user to find information even when he cannot precisely specify the object of his search. This process is illustrated in Chapter 2 by a query of a personnel database for a programmer with specific skills. Finding no one person who meets all of the requirements, the user of the system is able to enter a less restrictive query which produces a graphical data space through which he can browse in order to find an approximate fit to his requirements. The ability to generate such graphical data spaces from symbolic data bases allows this type of graphical access without the user having to enter the graphical data each time.

The desire to avoid specially constructed hardware led to the consideration of several alternatives. One of these was implementing a three dimensional information space [CCA]. Such a space would allow a much larger amount of information to be stored in a universe of modest dimensions, and lent itself to implementation on relatively inexpensive hardware. Such hardware has the capability of displaying "wire frame" projections of a three-dimensional model in real time. The user would fly a "spacecraft" among the various 3D data objects.

The drawbacks of this scheme were illustrated by the MIT experience with two-dimensional data spaces which showed the importance of color and shape to avoid getting lost in the graphical data space. The problems of the monochrome nature of the 3D display were further compounded by the absence of inexpensive (<$1M) means of displaying only the visible surfaces of objects, making them extremely difficult to identify from varying viewpoints.

Fortunately, the commercial display market advanced sufficiently quickly to enable the procurement of a display with functionality similar to that of the MIT system. Details of the hardware and the process by which

Figure 1.4


To Arpanet & Datacomputer
LA180
printer
programmer terminals
RP06
RP04
88 mb disk
TU16
9 track tape
PDP 11/70
768Kb core memory
LPS11
joysticks
control
vidicon
videodisk
videotape
RGB video
Lexidata display
sound
video switching

it was selected can be found in [HEROT et al]. Figure 1.4 shows the hardware configuration of the CCA prototype. The equipment shown by dashed lines is planned for future addition to the system.

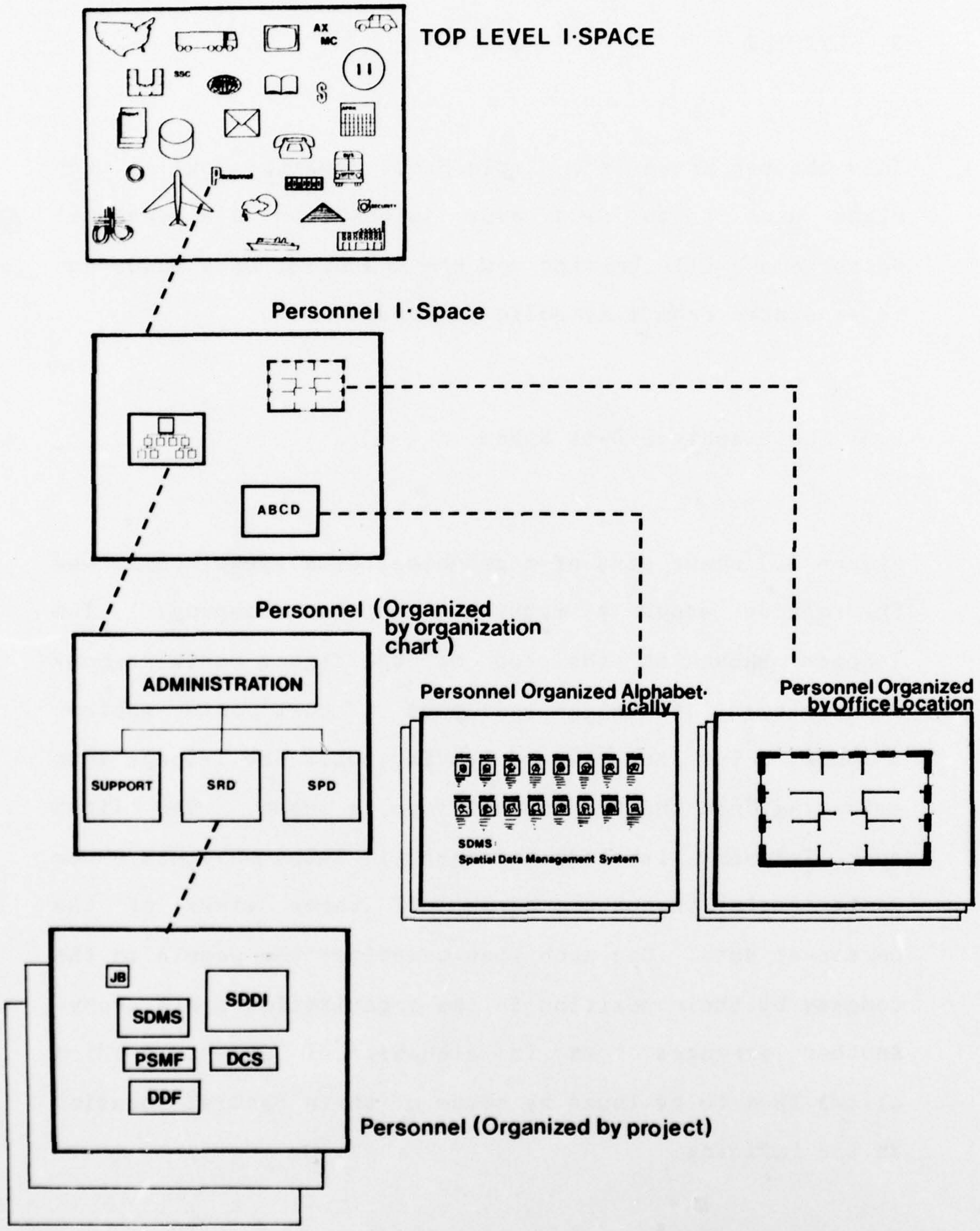
2. EXAMPLE

This chapter presents a sample SDMS, showing how a user might use it to find some information in a personnel database and illustrating how the Graphical Data Space can be generated from a symbolic database.

2.1 The Graphical Data Space

Figure 2.1 shows part of a graphical data space containing information about a mythical computer company. The I-Space shown at the top of the figure contains many icons, each of which indicates a port onto another I-Space. For the sake of brevity, only the I-Space tree emanating from the "Personnel" icon is shown. The first such I-Space, labeled ("Personnel I-Space"), has three ports, giving the user a choice of three views of the personnel data. One such view organizes the people in the company by their position in the organization's hierarchy. Another arranges them in alphabetical order. A third allows them to be found by means of their spatial location in the building.

Example Graphical Data Space Figure 2.1


TOP LEVEL I·SPACE
AX
MC
SSC
Personnel
SECURITY
Personnel I·Space
ABCD
Personnel (Organized by organization chart)
ADMINISTRATION
SUPPORT
SRD
SPD
Personnel Organized Alphabetically
SDMS
Spatial Data Management System
Personnel Organized by Office Location
JB
SDMS
SDDI
PSMF
DCS
DDF
Personnel (Organized by project)

These three I-Spaces were all generated from the same

Symbolic Data for Example Graphical Data Space Figure 2.2

Personnel (<u>name</u>, project)

Divisions (division, project, manager)

Projects (project, manager, nemployees)

Offices (<u>name</u>, number)

office-loc (<u>number</u>, x-coord, y-coord)

phone (<u>name</u>, ext)

home (<u>name</u>, address, city, state, zip, phone_number)

login-name (name, system, login-name)

languages (<u>name</u>, language)

systems (<u>name</u>, system)

---

symbolic data, shown in Figure 2.2. The Icon Definition Language statements which generate the I-Spaces is given in Appendix A.

Each of the three I-Spaces are made up of three planes of image data, permitting the user to examine the personnel data at several levels of detail. A scenario in which a user examines them is described below.

2.2 Search

Searches of the Personnel data can be made by examining the graphical data space, by issuing symbolic queries to the database management system, or by a combination of the two.

2.2.1 Graphic Search

Figures 2.3 through 2.15 illustrate a user's search through the Graphical Data Space of Figure 2.1.

Upon entering SDMS the user sees the scene depicted in figure 2.3. The top frame shows what is seen on the primary display, which at this point is a portion of the top-level I-Space. The middle frame shows the view on one of the auxiliary displays - a "world-view" of the entire top-level I-Space. The colored rectangle indicates which portion of the I-Space is currently visible on the primary display. The bottom frame is the second auxiliary display, which is presently blank.

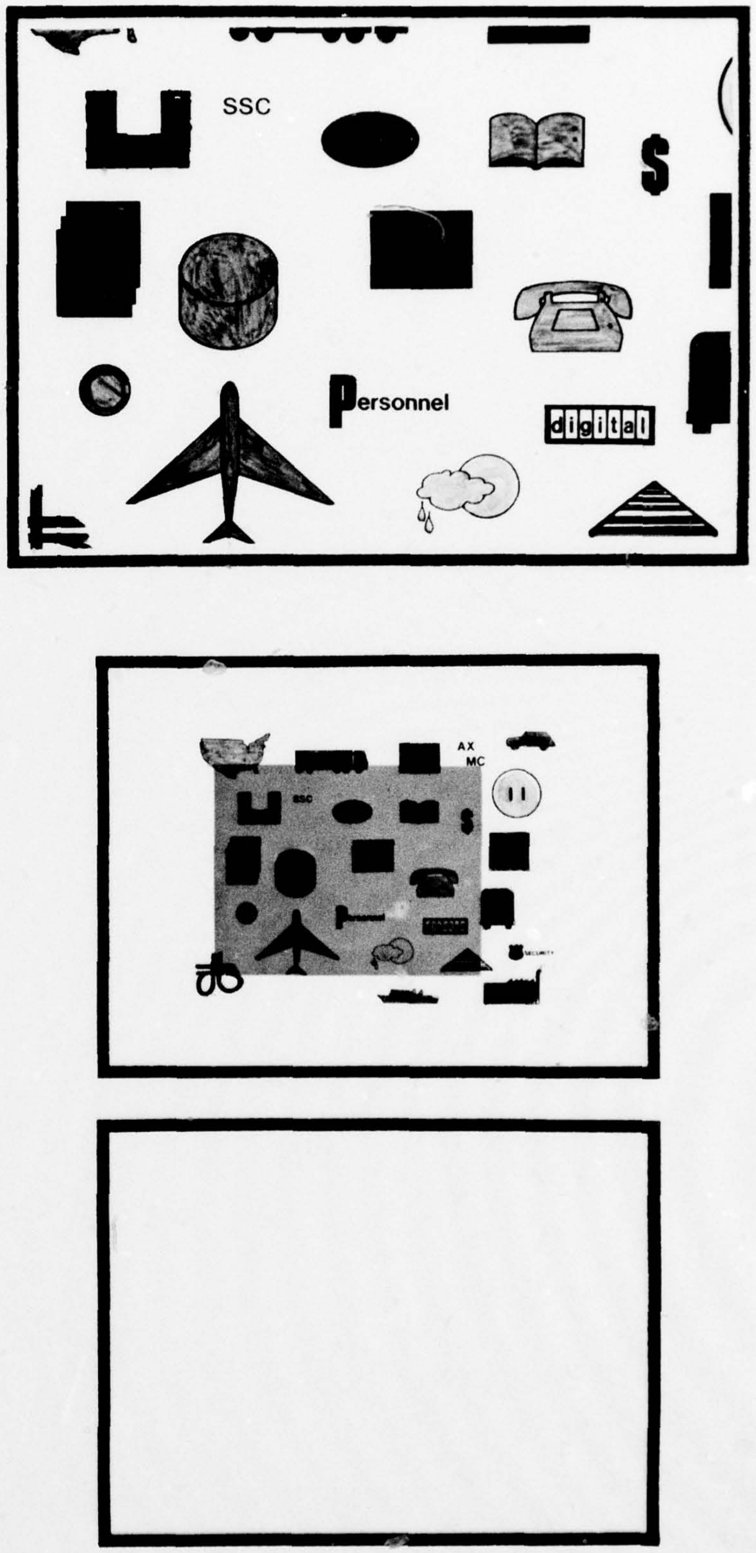
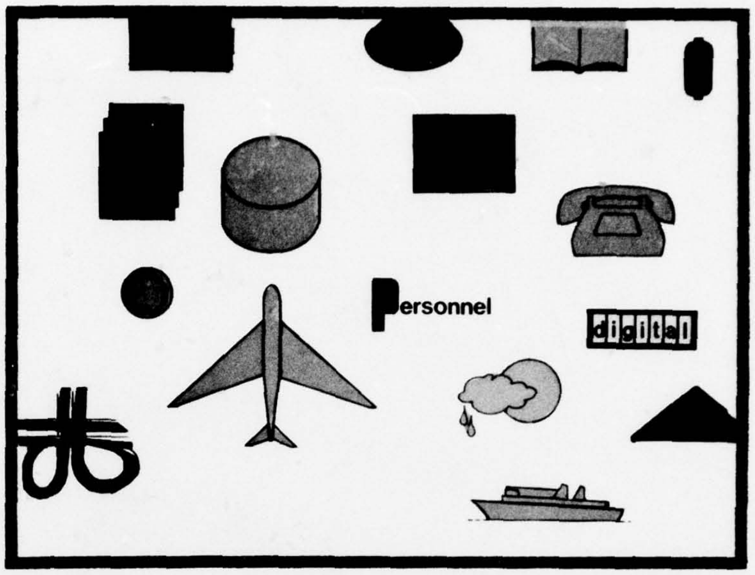
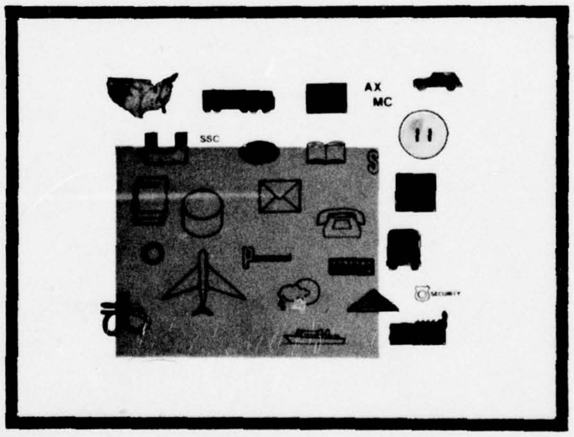


Figure 2.3

Our mythical user decides to look at personnel information. Seeing the "Personnel" icon in the lower left hand corner of the display, he uses the joy stick to move his viewport over that icon. As shown at right (Figure 2.4), the colored rectangle moves as the viewport moves, showing the user where he is in the I-Space.





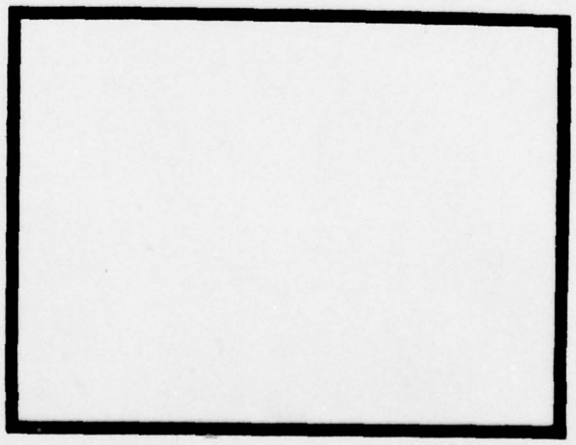
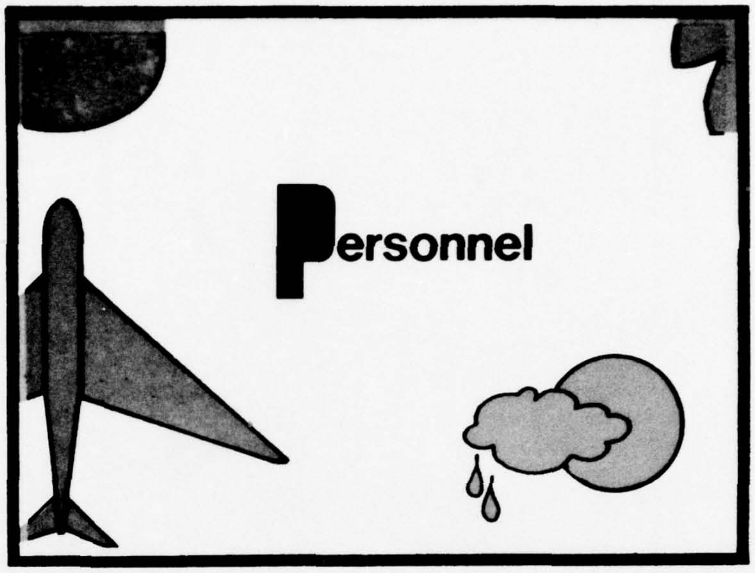
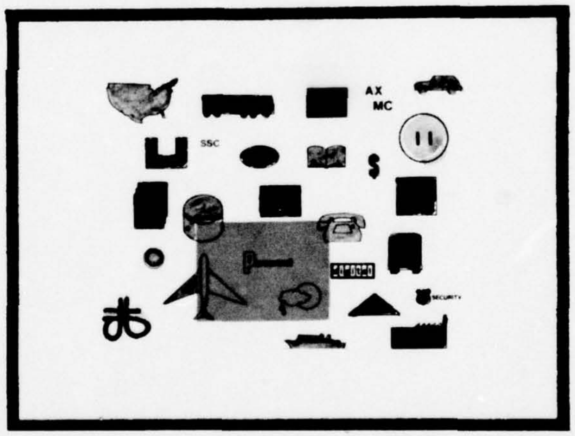


Figure 2.4

Having centered the Personnel icon, the user zooms in on it, resulting in the view shown at right. While the icon is becoming larger, the colored rectangle on the world view map is shrinking, reflecting the fact that a smaller portion of the I-Space is being viewed.





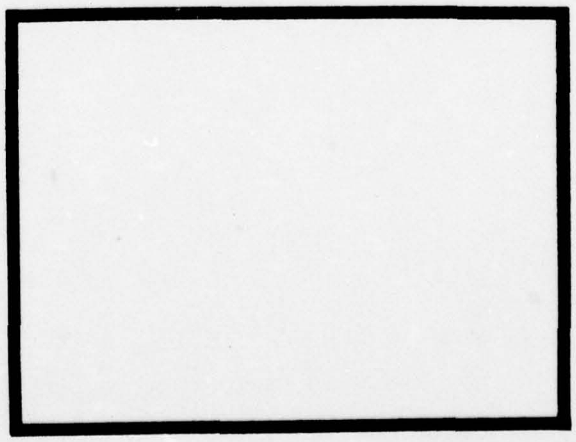


Figure 2.5

The zooming process results in the user "popping through" to a new I-Space, a much simpler one with only three icons. (Shown at right) These give him a choice of three ways of looking at the personnel data:

1. Hierarchical
2. By Office location
3. Alphabetical

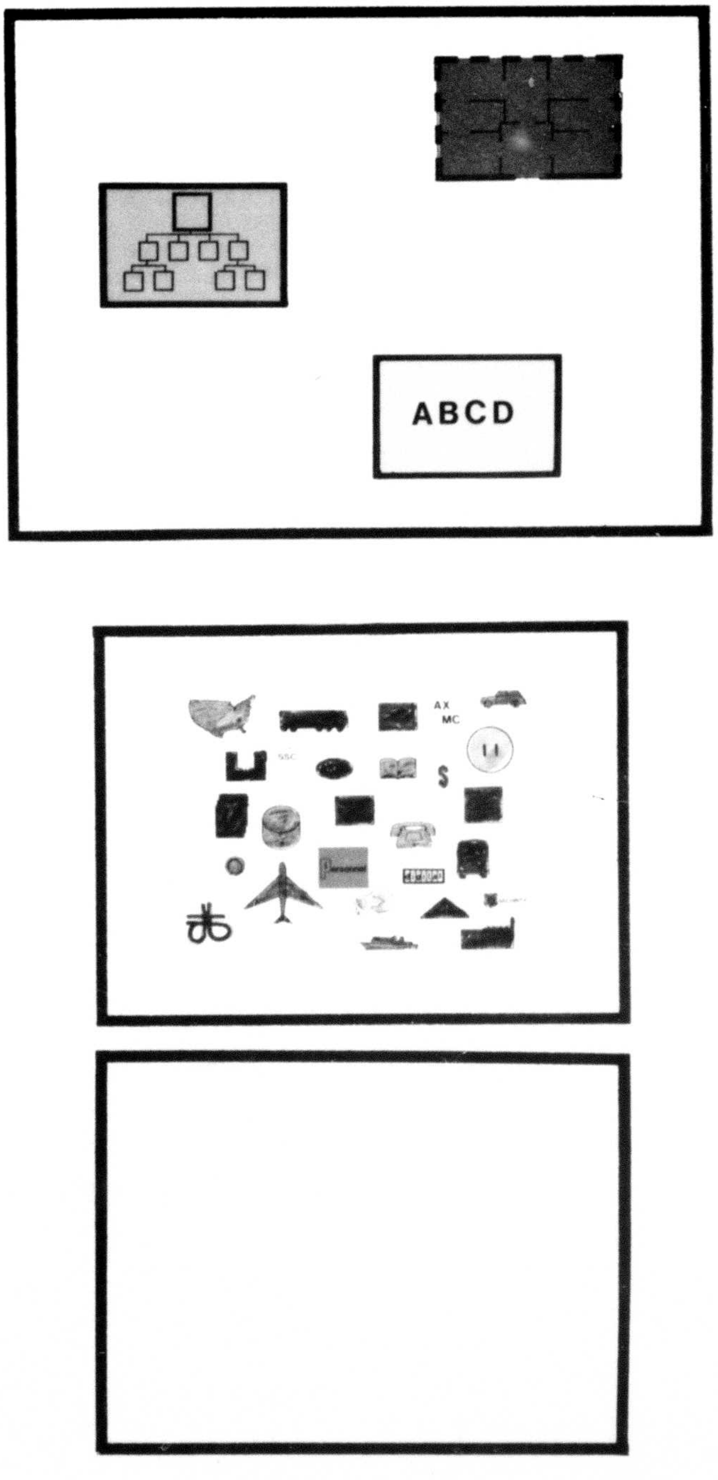


Figure 2.6

Selecting the hierarchical ordering, the user centers the corresponding icon on the screen (Figure 2.7) Now he zooms in on the icon (Figure 2.8), bringing him into yet another I-Space (Figure 2.9).

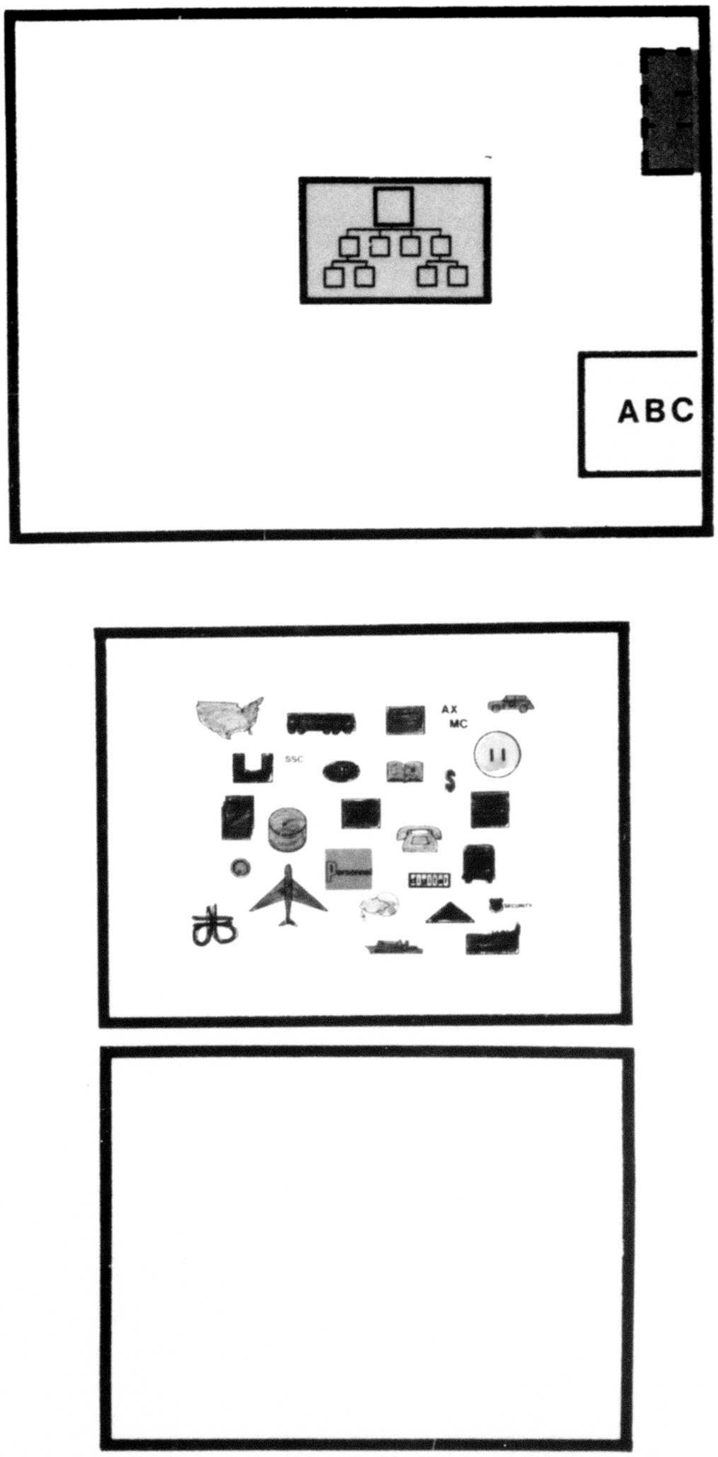


Figure 2.7

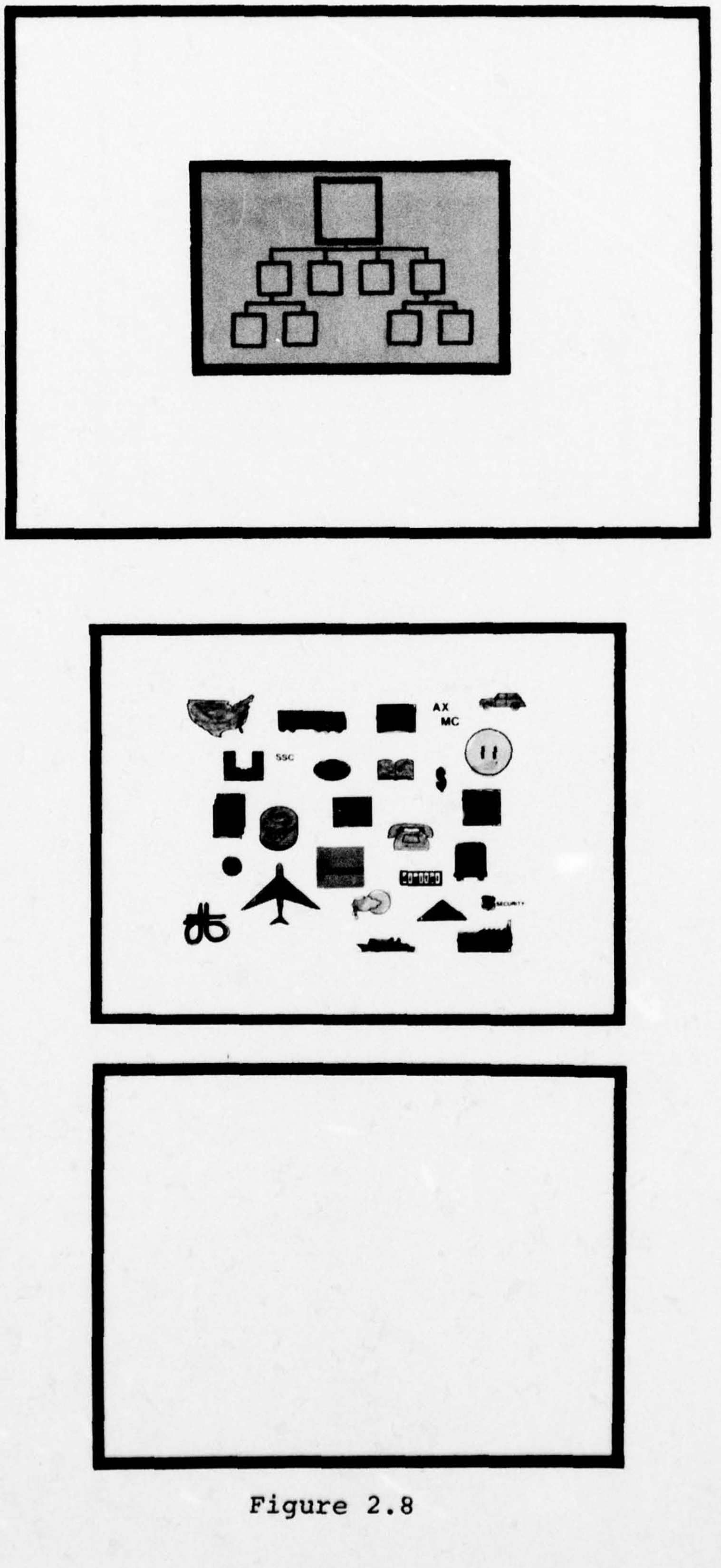


Figure 2.8

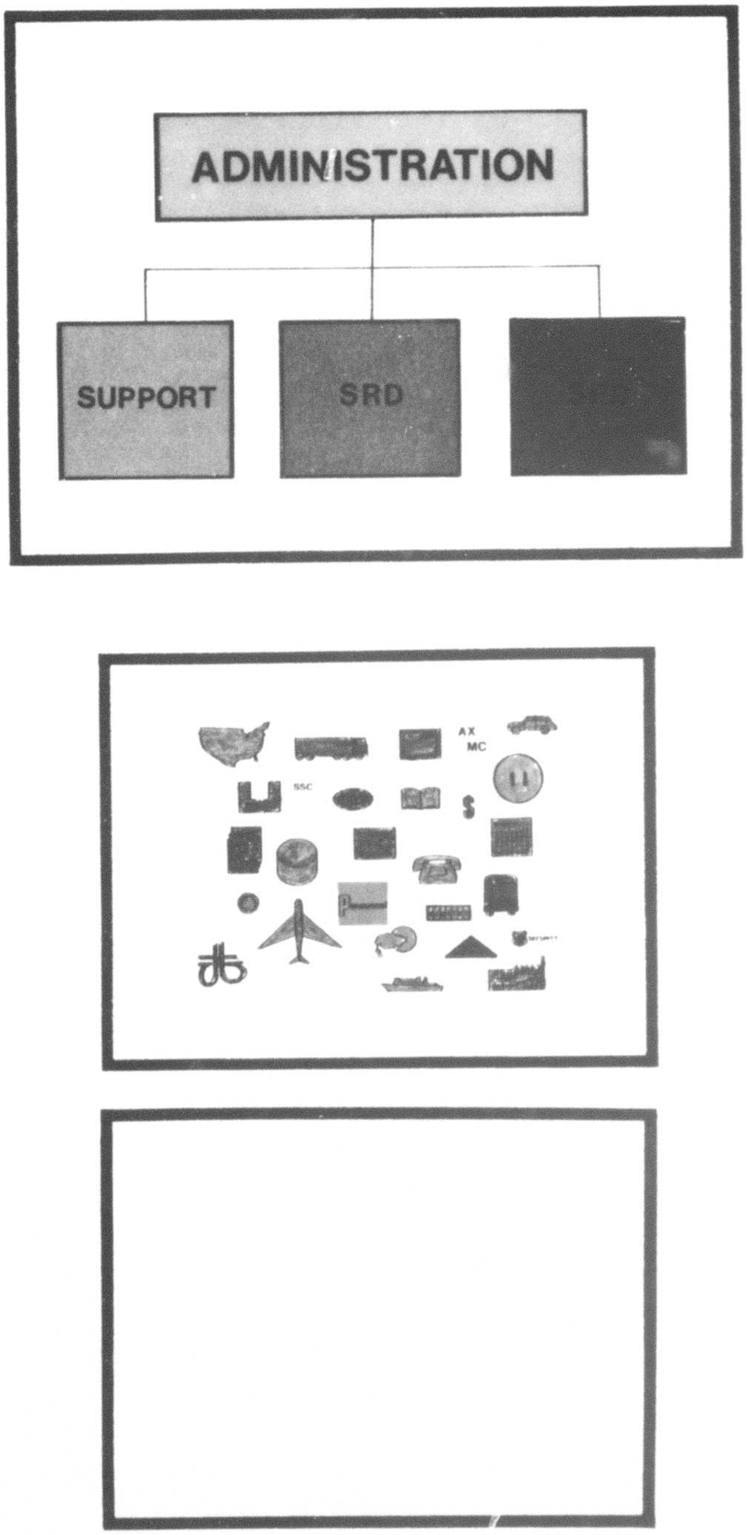
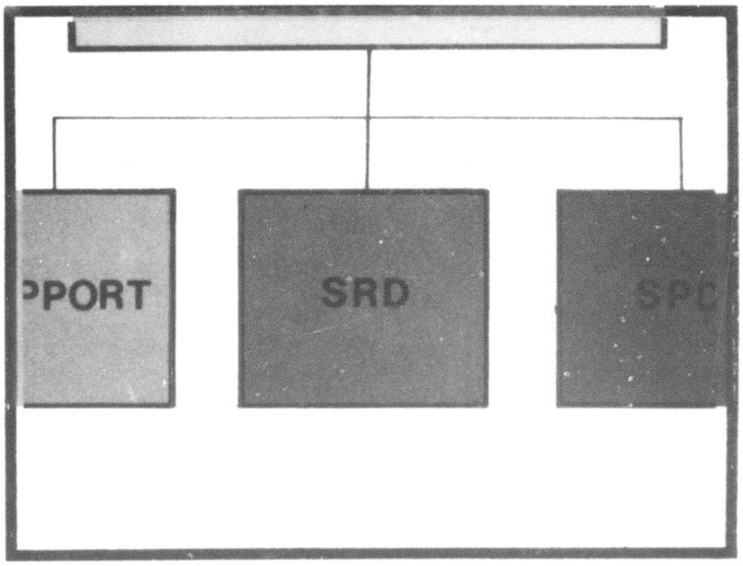
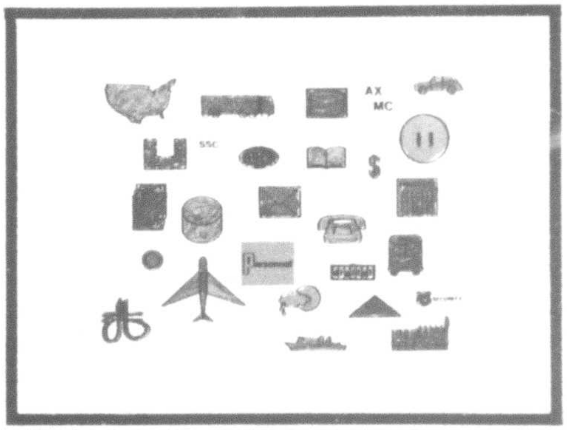


Figure 2.9

At this point, he decides to look at the Sponsored Research Division, indicated by the initials "SRD", so he zooms in on that (shown at right), and arrives at the I-Space shown in Figure 2.11 on the next page.





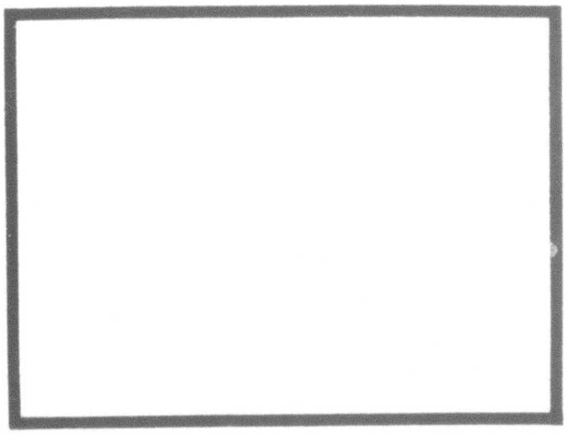


Figure 2.10

This new I-Space is a bit different from those seen previously. It has a world-view map defined for it, which appears on the second auxiliary display. This I-Space also has several planes of data, so that the user can zoom in for more detail without popping through to another I-Space.

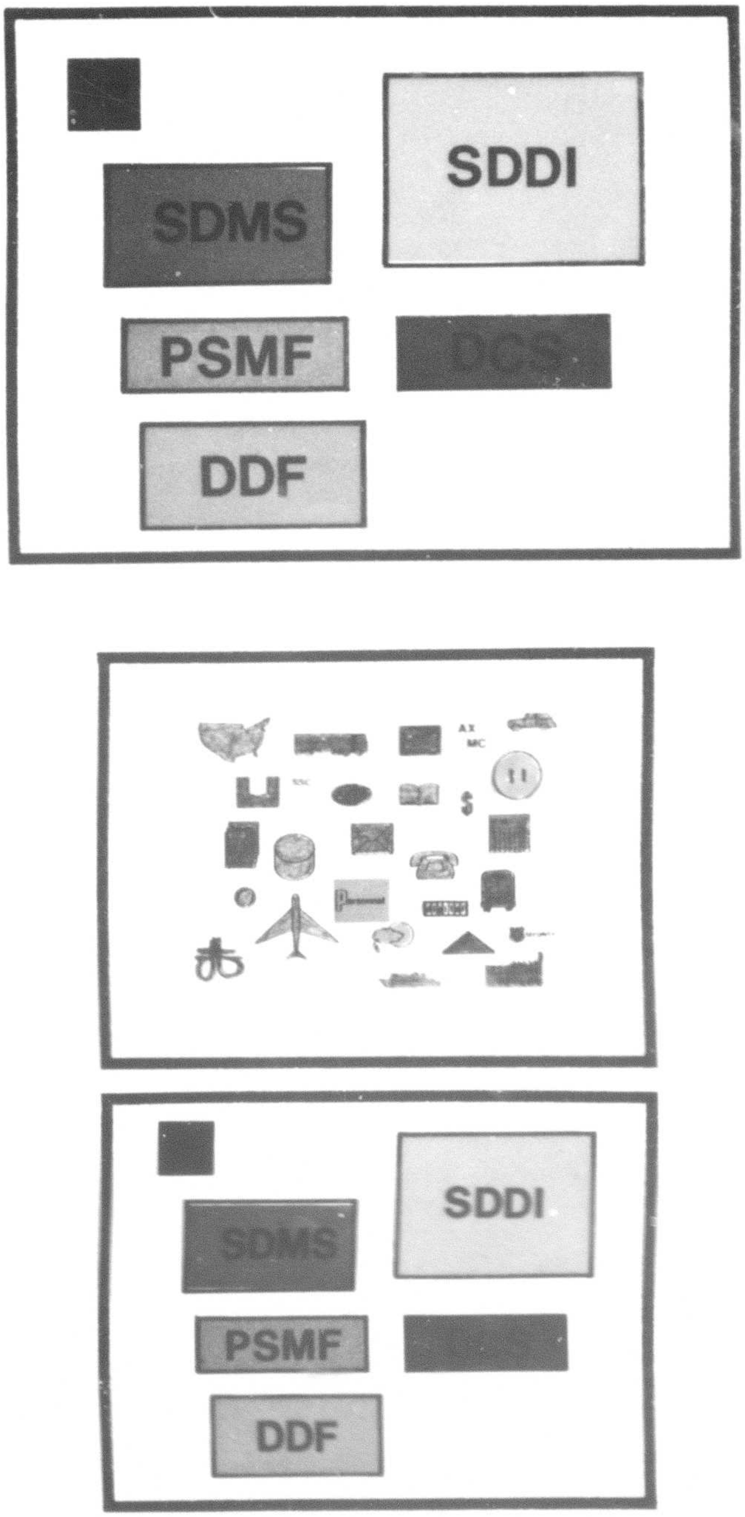
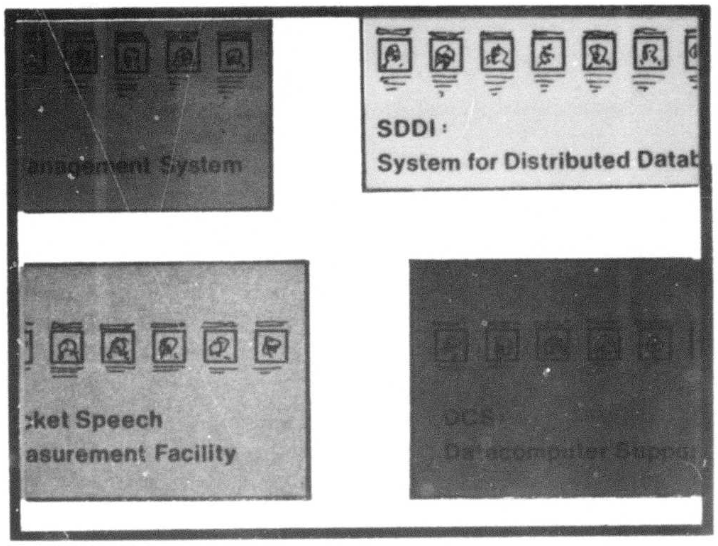
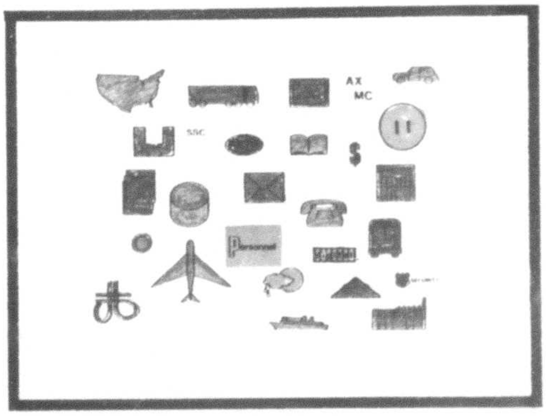


Figure 2.11

Zooming in on the I-Space produces the view at right. The result is different in some ways to that of previous zooming actions. In the three previous cases which brought us from the top-level to the level in the previous picture (2.11), each zoom carried us into a new I-Space. It was not possible to move between I-Spaces except by zooming and popping. In contrast, this last I-Space (Figures 2.11 through 2.15) allows the user to move freely between projects. Each of these projects is indicated by a color overlay and has a





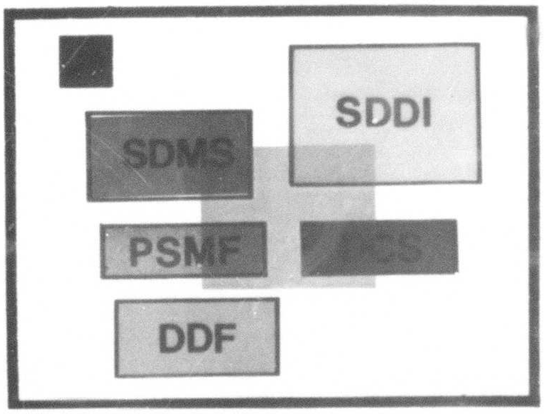
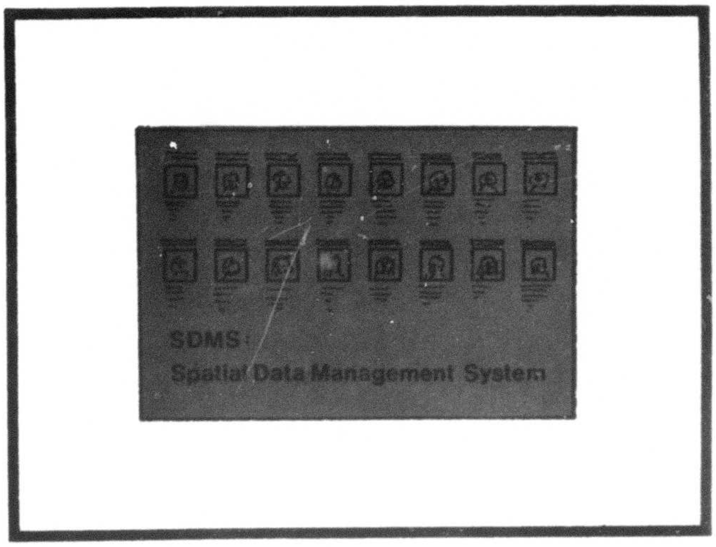
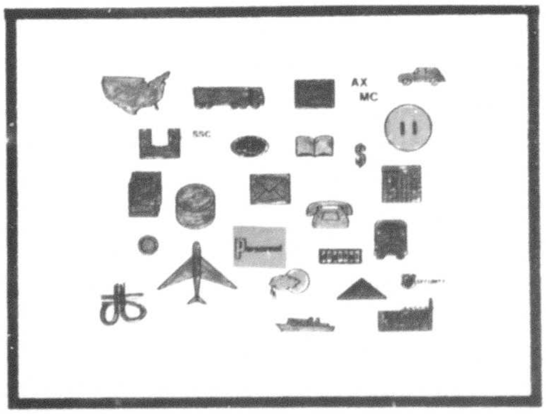


Figure 2.12

separate icon to help locate it, but they are all part of the same I-Space. The decision of which way the Graphical Data Space is set up is made by the Database Administrator when the I-Spaces are created.

Deciding to look at the SDMS project, the user centers its icon on the screen (Figure 2.13). As he does so, the colored rectangle on the second auxiliary display moves to show his new location in the I-Space.





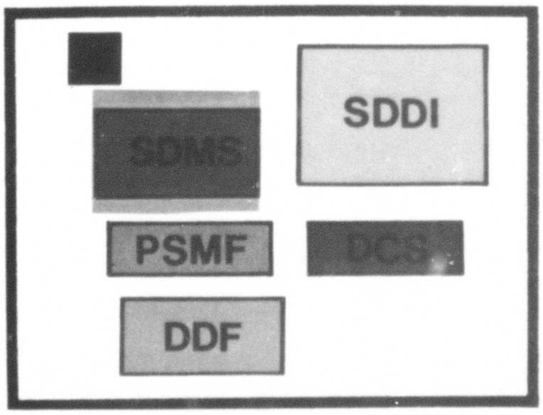
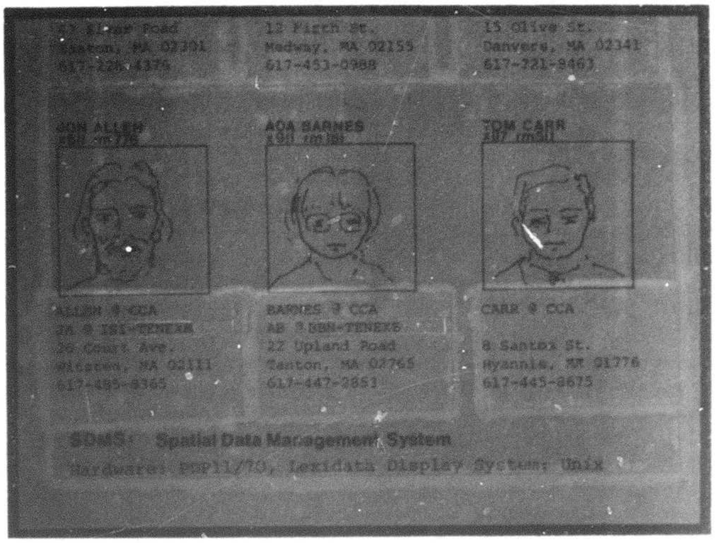


Figure 2.13

Next, the user zooms in on the SDMS Icon. This action causes more detail to appear, in the form of text. The colored background reminds him of which icon he is examining, as does the (now smaller) colored rectangle on the second auxiliary monitor.



[illegible] Road
[illegible], MA 02301
617-226-4376

12 Firth St.
Medway, MA 02155
617-451-0988

15 Olive St.
Danvers, MA 02341
617-721-9463

[illegible] ALLEN
[illegible]

ADA BARNES
[illegible]

TOM CARR
[illegible]

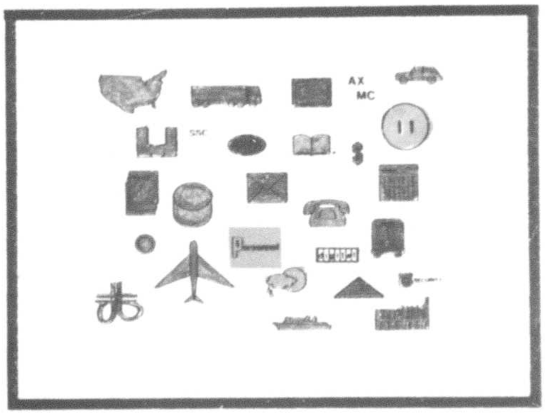
ALLEN @ CCA
JA @ ISI-TENEXA
26 Court Ave.
[illegible], MA 02111
617-495-8365

BARNES @ CCA
AB @ BBN-TENEXB
22 Upland Road
Tanton, MA 02765
617-447-2851

CARR @ CCA

8 Santex St.
Hyannis, MA 01776
617-445-8675

SDMS: Spatial Data Management System
Hardware: PDP11/70, Lexidata Display System; Unix



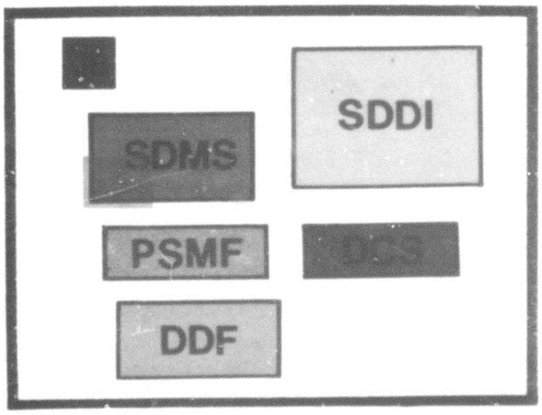
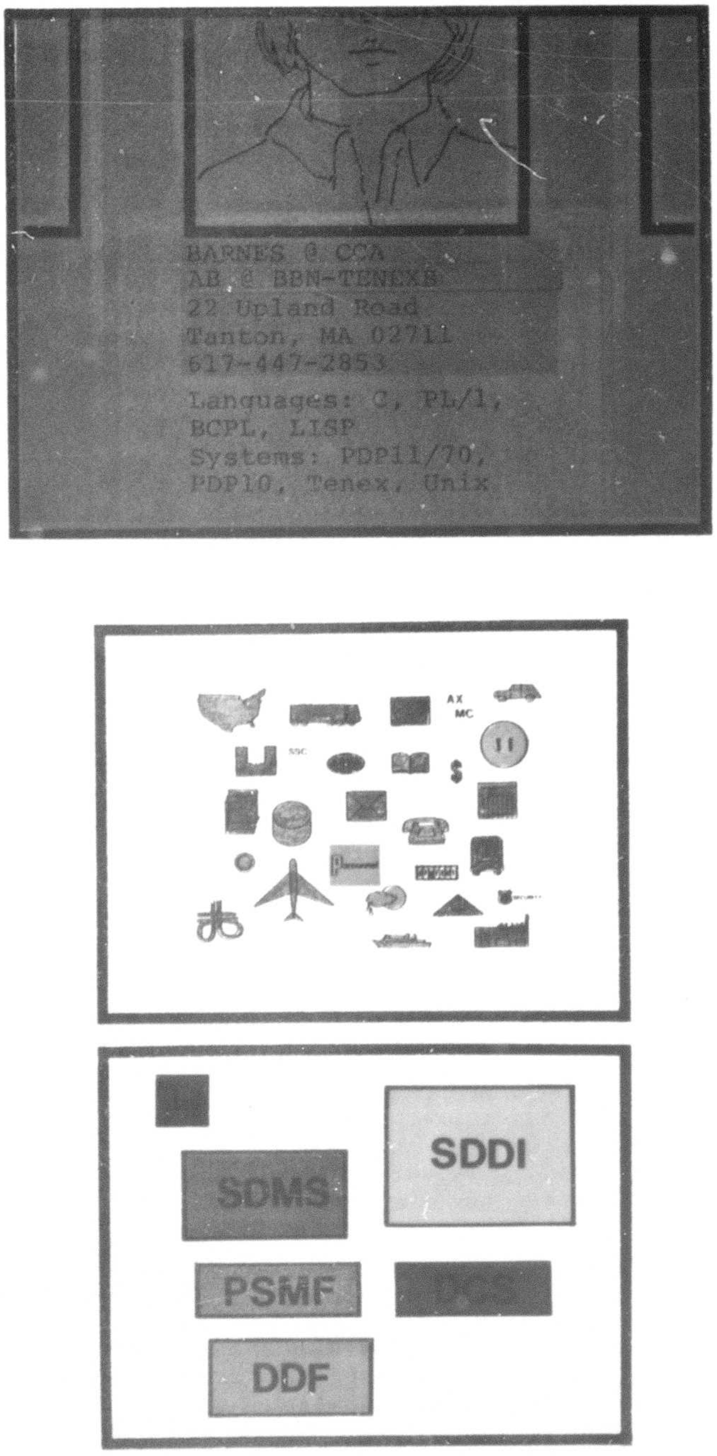


Figure 2.14

Zooming in still further causes still more text to appear (shown at right).

As we have seen, the user can peruse this database of personnel at any desired level of detail, making his own decision about the optimum trade-off between the amount of information available and the space which must be traversed in order to see it.



BARNES @ CCA
AB @ BBN-TENEXB
22 Upland Road
Tanton, MA 02711
617-447-2853
Languages: C, PL/1,
BCPL, LISP
Systems: PDP11/70,
PDP10, Tenex, Unix

Figure 2.15

Unfortunately, this particular query does not turn up any people with all of the required skills. Trying for a close approximation, our user can combine graphical and symbolic search. First he formulates a query which retrieves all programmers with a knowledge of any of the desired languages:

```
range of P is PERSONNEL
range of S is SYSTEMS
range of L1 is LANGUAGES

blink P
   where (P.name=S.name and
          S.system="PDP-10" and L1.name=S.name and
          ( L1.language="LISP" or
            L1.language="SNOBOL" or
            L1.language="PL/1" ))
```

The use of the blink command causes the icons of the selected programmers to blink, making it easy to scan the graphical data space for the relevant personnel and use the data found with each one to help make a decision. The search could have been further simplified had the user been willing to allow the extra time required to generate a new I-Space, which he could have done by specifying the restriction:

2.2.2 Symbolic Search

Let us suppose that our user wishes to find a programmer with experience on a PDP10 and a knowledge of LISP, SNOBOL and PL/1. While he could search through the graphical data space looking for such people, a more efficient way would be to use the symbolic data base. The following statements* accomplish the task:

```
range of S  is SYSTEMS
range of L1 is LANGUAGES
range of L2 is LANGUAGES
range of L3 is LANGUAGES

retrieve into FOUND (name=S.name)
    where (S.system="PDP-10" and
           L1.name=S.name and L1.language="LISP" and
           L2.name=S.name and L2.language="SNOBOL" and
           L3.name=S.name and L3.language="PL/1" )
```

---

* The statements shown here are in an augmented version of QUEL, the query language of the INGRES relational database management system.

```
range of P is PERSONNEL

range of S is SYSTEMS

range of L1 is LANGUAGES

associate P
   where (P.name=S.name and
          S.system="PDP-10" and L1.name=S.name and
          ( L1.language="LISP" or
            L1.language="SNOBOL" or
            L1.language="PL/1" ))
   with ISPACE
```

The I-Space named ISPACE must have been created previously. This association is used temporarily. The user can go to ISPACE and browse through the icons in it, knowing that they all represent programmers that have PDP-10 experience and have worked with one of the three desired languages. Such use of the Icon Class Description Language is illustrated in Appendix A. After the user has found the programmer he wants, he can break the temporary association with:

```
break association of PERSONNEL with ISPACE
```

The association is forgotten and the icons that were created because of the association are erased.

3. SYSTEM SPECIFICATION - OUTPUT

This section describes those actions which the user can perform in order to search the SDMS Graphical Data Space (GDS) and retrieve the data found there. It is important to note that these activities do not include querying the symbolic database. While such queries are permitted (and are described in Chapter 8), most often the user will interact with a graphical _counterpart_ to symbolic data stored in the DBMS.

3.1 System Environment

The Spatial Data Management System runs as a user program under the Unix operating system. It consists of one or more processes which interact with the user and manage the SDMS resources. Some of these programs update the displays to give the effect of motion through the various Information spaces (I-Spaces). Others allow the user to make queries of INGRES, a relational database system. In addition, there are processes which operate in the background, unseen by the user, performing operations which update the symbolic database (INGRES), cause changes in the symbolic database to be reflected onto the graphical data space, and receive messages, indicating them in the GDS.

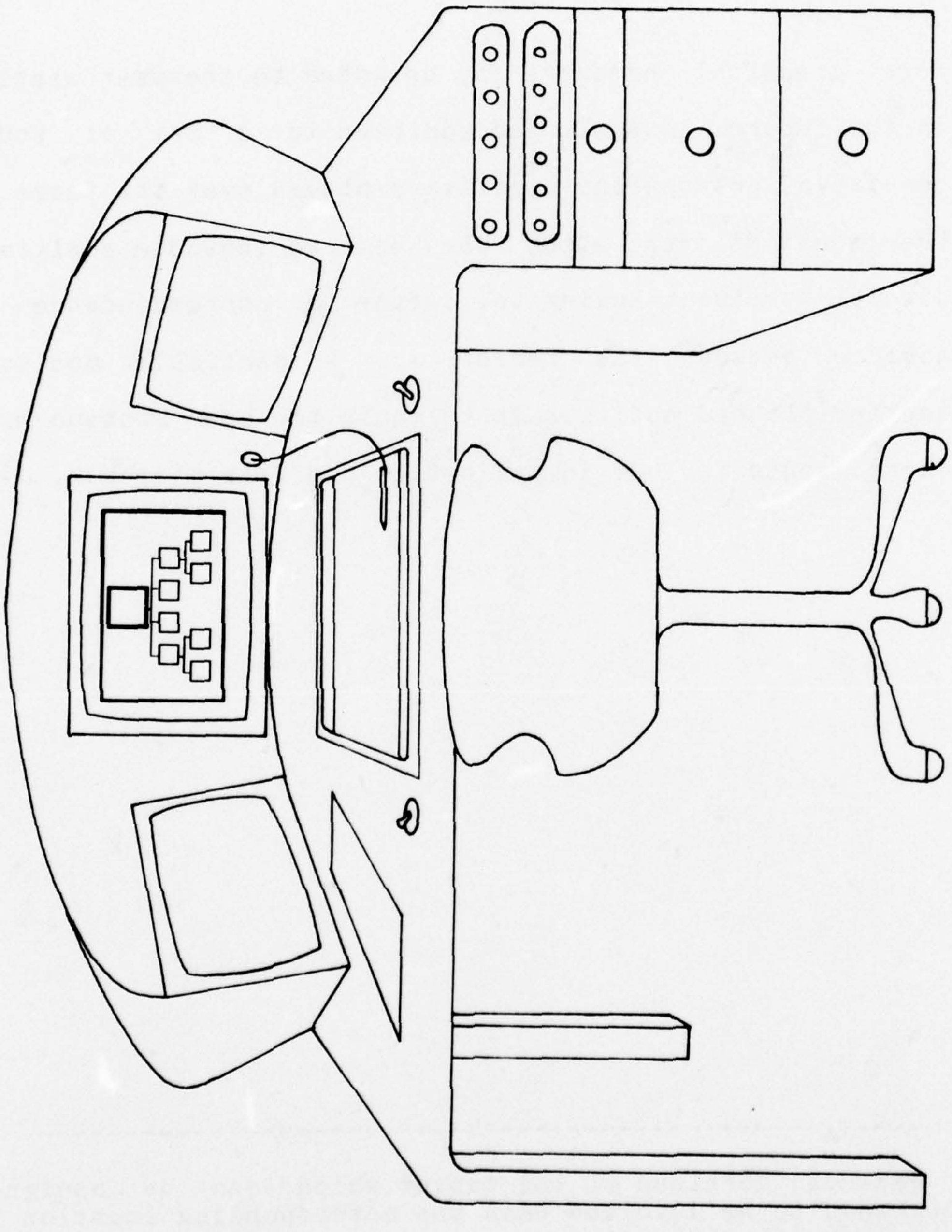
A user enters the SDMS by typing the command "sdms" to the Unix shell. If he has used the system before, he is positioned to the spot in his personal data space where he was when he last used the system. If he is a new user, he is positioned at the top level of a default data space.

3.2 The User Station

The planned configuration of the user's work station is shown in Figure 3.1. It consists of three color television monitors, two joy sticks, a data tablet, and an alphanumeric keyboard. The central monitor is used for interacting with the Information space. The two joy sticks control the motion of the observer within the I-Space, determining what is shown on the central monitor. The two auxiliary monitors serve as navigational aids, showing the user where he is in the SDMS world. The tablet is used for pointing to objects, adding to the

User Work Station

Figure 3.1



graphics on an image plane, and issuing commands by means of menus.* The keyboard is used for entering text and for issuing commands to the operating system.

More graphical hardware may be added to the user station in the future. One planned addition is a set of touch sensitive, transparent digitizers placed over the faces of the monitors, to allow the user to indicate positions directly, without having to define a correspondence in advance between the tablet and a particular monitor. Another planned addition is to equip the user station with stereo sound for use in navigation and data playback.

---

* Reserved portions of the tablet which can be assigned commands to be involved when the corresponding location is touched with the stylus.

3.3 Motion in the Plane

The right hand joystick controls motion in the current image plane. Pressing it to the right causes the position of the window being viewed of the image plane to move to the right. The effect on the display is that the various icons move to the left and off the screen as new ones appear on the right. Similarly, pressing the joystick away from or towards the user causes apparent motion in the vertical plane. All of this is accomplished by moving strips of byte-per-point data describing the image plane from secondary storage (in this case a DEC RP04 disk), through the host computer to the display. For the planned screen resolution of 480 lines of 640 pixels, a screenful of data can be paged across the screen in a little over one second [HEROT et al]. Faster times can be achieved by operating at a lower resolution, an approach that offers other advantages outlined below.

As the user moves over the i-plane, his position is continually indicated on a world-view map on one of the auxiliary monitors. This map is a low resolution image of the entire I-Space, which shows principal landmarks and the user's current position. The user's position is indicated by a transparent rectangle which precisely indicates what portion of the I-Space is currently being shown on the main display. This map can be used at any time to achieve rapid transit to any point in the I-Space. By indicating the desired position with the data tablet (or merely touching the screen when the touch sensitive digitizer is present), the user can be quickly transported to the desired location without the necessity of traversing all of the intervening space.

When sound is added to the system, some icons will emit a characteristic sound to aid in finding them. As such an icon goes off the screen, the sound will appear to come from that direction, with its volume inversely proportional to its distance from the observer.

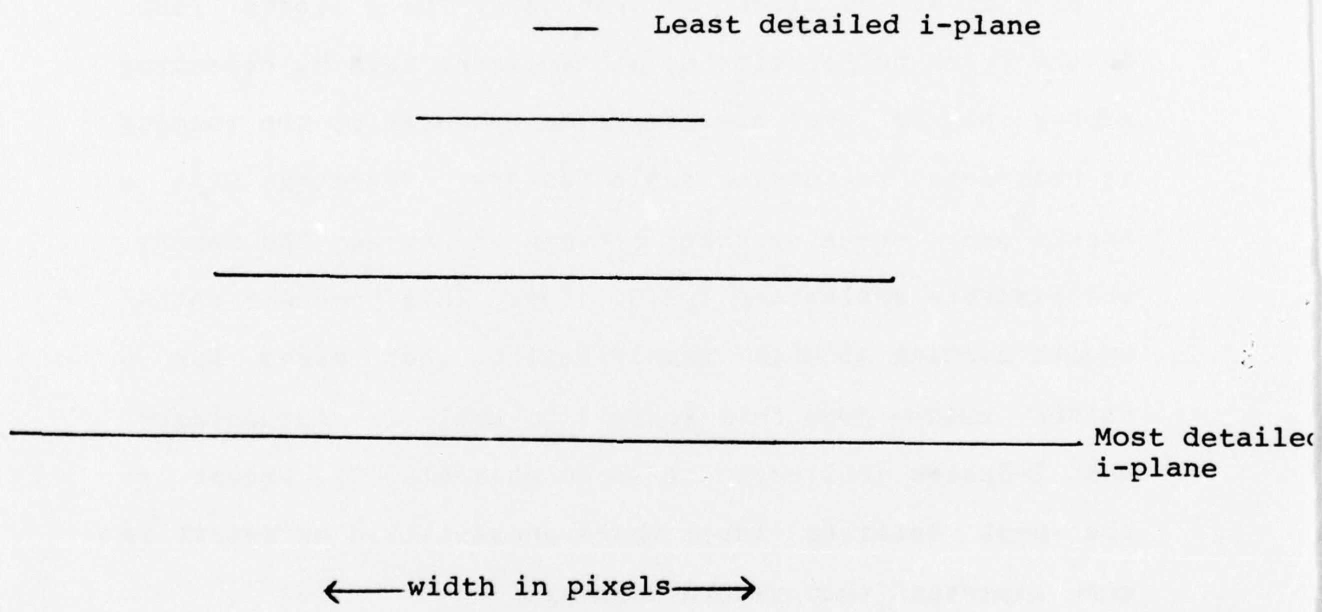
3.4 Zooming within an I-Space

The left-hand joystick controls the apparent distance of the observer from the surface of the I-Space. Pressing it forward gives the effect of descending for a closer look. As the frame buffer display accomplishes this by repeating memory pixels over successive screen pixels, the zooming is restricted to integer scale factors. Starting with a one-to-one correspondence between the screen and memory, the possible scales are 1,2,3,...16. This produces rather smooth zooming at high magnification, but makes for a rather sudden jump from scale 1 to scale 2. Accordingly, most I-Spaces will never be shown at scale 1, except at the most detailed level where presentation of detail is more important than smooth zooming.

For each i-plane in an I-Space, except for the most detailed one, there is a scale defined at which more detail is introduced. This is done by reading from the disk a new array of image data, called an <u>image-plane</u> or <u>i-plane</u>. An I-Space can be constructed out of one or more of these i-planes. An i-plane is thus a complete raster-scan image of its associated I-Space at some defined level of detail. The addition of detail by zooming is accomplished by replacing the zoomed-in version of the i-plane with the full scale version of the new,

---

Cross Section of i-planes in an I-Space Figure 3.2



---

higher resolution i-plane (see figure 3.2). The actual scale values at which inter i-plane transitions occur will be determined by experimentation and will be specifiable by the database administrator. In any event, the transition is orchestrated to give the appearance of continuous addition of detail by bringing the new i-plane into the frame buffer while the old one is in the process of zooming. At the moment when the user would expect the old i-plane to expand, the new, more detailed i-plane is displayed.

The zooming effect is indicated on the world-view display by changes in the size of the transparent rectangle. As the user zooms in, the rectangle becomes smaller, and vice-versa.

Motion within an I-Space is homogeneous. The lateral motion of the user in any i-plane causes corresponding motion in the I-Space proportional to the scale of the i-plane.

3.5 Interaction with Icons

The incremental addition of detail allowed by zooming is usually used to help the user find the information he seeks at a coarse level and to zoom in to find out more specific information at the appropriate place. In the case of a photograph, this happens when each i-plane contains a more detailed version of the image. With text, it happens when the icon for a document is replaced by the actual text.

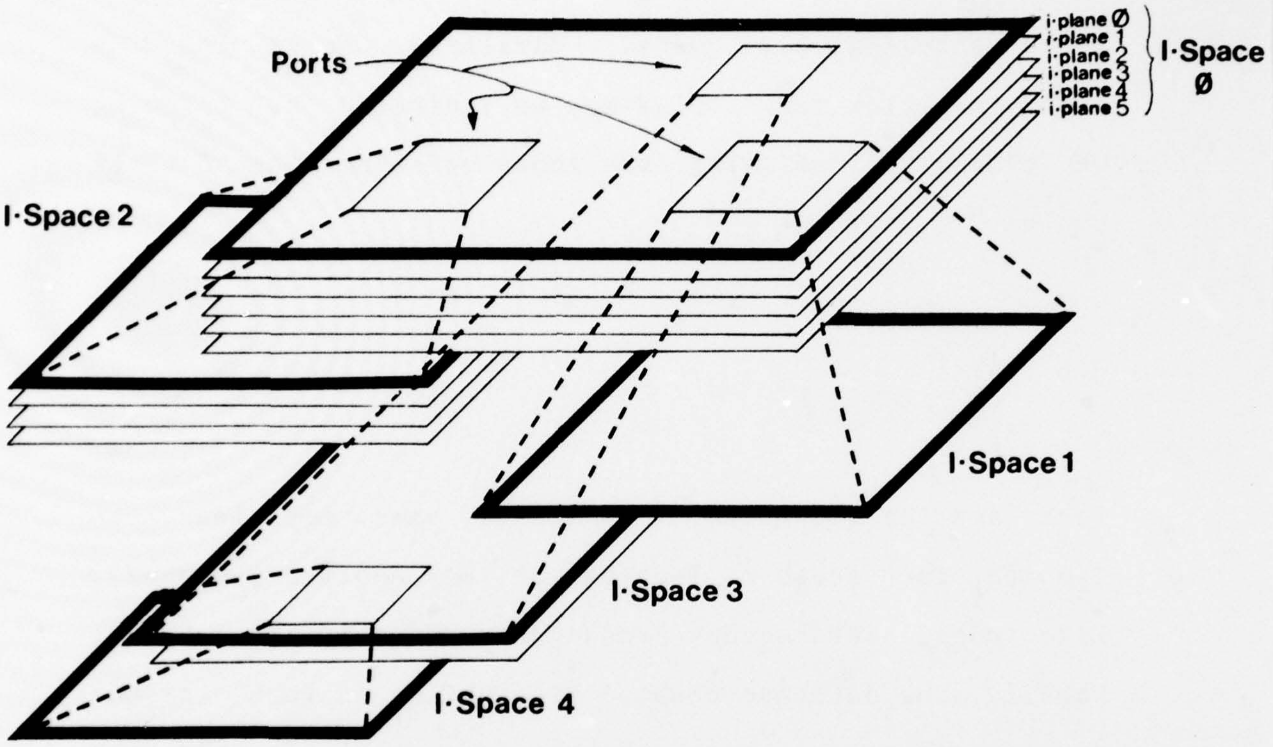
It is also possible for the icons themselves to carry the desired information. In a database of ships, the color, size and shape of the icon can indicate its nationality, gross tonnage, and class. As the user zooms in on the icon, more features can be added. These features may have been entered manually by the person who drew in the icon, or they may be generated from attributes stored in a symbolic database.

Icons can serve as more than just visual indicators of the location of information. If they are associated with tuples in a database, they can be used to indicate those tuples for the purpose of formatting database queries. In order to ask for more information about a ship, the user may, instead of specifying its name, merely point at it while entering his query. Similarly, the results of a query involving many tuples may be indicated by blinking or otherwise modifying the icons corresponding to those tuples.

3.6 Ports

Ports are the mechanism by which a user may leave an I-Space for another I-Space or any arbitrary process. Like icons, ports occupy rectangular areas in the I-Space. Usually, the database creator will define an icon in the same position as a port, to make it possible for the user to find it. This is not required, however, so it is possible to have "secret" ports or ports defined on portions of photographs (see Figure 3.3).

Nesting of I-Spaces through ports Figure 3.3

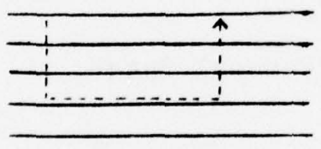


3.6.1 Inter-I-Space Transitions

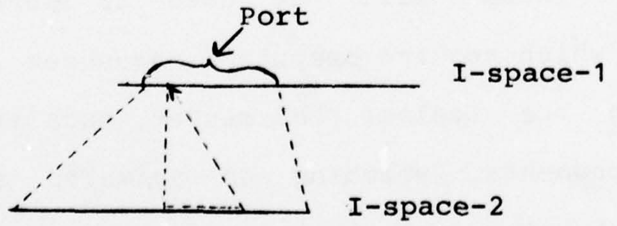
A port is <u>activated</u> by zooming in on it. The visual effect may be similar to zooming in on an icon, in that the image of the new I-Space which the user sees may be a higher resolution version of the last image of the old

I-Space. However, regardless of the motion the user takes within the new I-Space, upon returning, he will be at the same port. This is contrasted with zooming in on an icon, moving over to a new icon and backing up, which would leave the user looking at the second icon on the top level (see Figure 3.4).

Figure 3.4



(a) Motion within an I-space



(b) Motion between I-spaces

As ports allow the nesting of I-Spaces, some navigational aids would help the user find his way around. While the world-view map will always display the top-level I-Space, to permit rapid transit at any time, the third monitor, unused thus far, can display a world-view map for the current I-Space. Alternatively, the user can request that it display a diagram showing the hierarchy of I-Spaces along with his current position within it. This monitor is, in general, free to be used by various processes for their own purposes.

3.6.2 Virtual Terminals/Perusal Space

The ability to activate an arbitrary process provides a facility for implementing perusal spaces which can have quite different properties of motion and activity than I-Spaces. These will be used to perform various activities which require computer resources arranged on other than the typical SDMS manner, such as reading and writing documents, watching an animated sequence, or writing, debugging, and running Unix programs.

There will be a facility through which such programs may make use of the resources of the SDMS and the underlying operating system. In addition to the usual Unix system calls, there will be routines for manipulating ascii text on the display, for generating graphics, and for controlling special video and audio devices.

There will be at least four functions provided for running at virtual terminals which will make use of the above facilities. They are:

1. A text perusal and editing facility similar to the Rand editor (Ned)
2. A system for recalling video frames and sequences from a video disk.
3. A "virtual terminal" facility for assigning ports to arbitrary Unix processes.
4. A message system similar in functionality to MSG.

4. NAVIGATIONAL AIDS

The term navigational aids is used here to refer to those media which are used to help the user to orient himself in the database and to determine the direction he should take. Such aids include various maps of the universe of data and sound cues.

4.1 Maps

The MIT work has shown the importance of a map to achieving the feel of spatiality. While a user sitting at a single display which shows a window onto the I-Space eventually begins to think spatially and learns his way around, that process happens much more quickly when he can see where he is on a map of the entire data world. In fact, many users rely on the map almost exclusively for finding a location in the database. The main display then serves as an aid in recognizing icons and homing in on them.

This section describes several map facilities which can be called up on either of the two auxiliary monitors. The decision as to which choices will be offered to the user at which points in the database, along with appropriate default values, will be made after the system is operational and some experience has been gained in having people use the system.

4.1.1 The World-view

The world-view map presents a picture of the entire I-Space, upon which the user's current position is indicated. This map allows an additional form of motion through the database called rapid transit. By indicating the desired position, through the tablet or (future) superimposed touch sensitive digitizer, the user is moved to the top level i-plane at the indicated position without having to traverse the intervening data space. If the world-view map for the top-level I-Space is always maintained on one of the auxiliary monitors, even when the user is in a subsidiary I-Space, it is always possible to return quickly to that I-Space without having to back up through the intervening levels.

4.1.2 I-Space Hierarchies

When an SDMS contains nested I-Spaces, an <u>I-Space hierarchy</u> map can be called up to show the relationships among them. On such a map, each I-Space is represented by an icon, and possible transitions (from a port in one I-Space to the top-level of another I-Space) are indicated by arcs between them. Arcs indicating a parent-child relationship are indicated by heavier lines than those which are merely possible transitions. As with a world-view map, the user may employ rapid transit to move from one I-Space to another.

4.1.3 Program defined maps

Any perusal space (process activated by passing though a port) may define its own map. Planned implementations include:

1. A table of contents for document perusal
2. An "hourglass" to show the percentage of a video sequence which has been viewed
3. A status display showing which Unix processes are connected to a virtual terminal.

4.2 Sound

The SDMS will provide a facility for recording sound and attaching it to any point in an I-Space. Typically this space will be occupied by an icon and associated data. The stereo balance and volume of the sound will be a function of the distance of that point from the center of the screen. This facility will probably be most useful in locating icons that are just off-screen, since the sound will give the user an idea as to the direction in which to move. To counter the annoyance of constantly hearing the sound of an icon which has been found and moved onto the screen, an option will be provided which will turn off any sound whose source point has been on screen longer than a specified time.

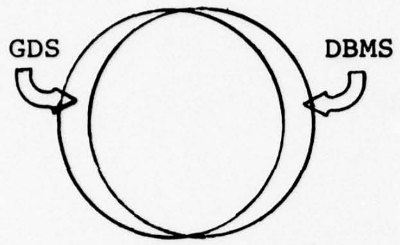
5. OVERVIEW OF THE GDS/DBMS INTERFACE

Much of the previous description has concerned the graphical data space (GDS) of SDMS. The concept of information spaces (I-Spaces) has been introduced along with the tools for "flying" about these I-Spaces. SDMS also has a companion Database Management System (DBMS) for symbolic data storage and manipulation.

The GDS and the DBMS can each represent data in a unique fashion. The GDS is oriented towards the storage of photographs, maps, diagrams and drawings. The methods of accessing data in the GDS are spatial in nature. Data is found by flying to it through the use of icons and navigational aides. This access method makes browsing through data a natural task. The DBMS of SDMS is a relational database system which allows the usage of symbolic relationships among data. Data is accessed by its attributes, not by browsing.

An important goal for SDMS is to provide facilities for representing data in both the GDS and the DBMS. The data in each may be distinct or it may be shared.

Figure 5.1



The data in both circles constitutes all the information in an SDMS.

Shared data has an important advantage. It may be accessed and modified by either standard DBMS queries or by the spatial methods which are unique to an SDMS system. This section will describe the basis of this dual representation and the manner by which it is achieved.

In the CCA implementation of SDMS, the DBMS used is INGRES, a relational database system [HELD STONEBRAKER WONG]. The query language of INGRES is called QUEL. Although the concept of SDMS is independent of the choice of DBMS, the specification of the interaction between SDMS and its DBMS must necessarily involve specifics of the DBMS. Section 5.1 contains a description of INGRES and QUEL. Those readers who are familiar with INGRES may wish to skip to Section 5.2.

5.1 INGRES - the underlying DBMS

INGRES is a relational database system that runs under the UNIX operating system on the PDP-11 machine. Briefly stated, a relation is a set of data elements where each element is a tuple of n values. Letting r be a tuple, we can represent it as:

r = (r1,r2,...,rn)

where each ri belongs to some domain which will be called Di. A relation may be thought of as a table. Each row of the table contains one tuple. Each column is called an attribute. The set of values in the i-th column are the i-th values of each tuple in the relation. So the values in the i-th column are all from the domain Di. The relation is truly a set of tuples, for there are no duplicate tuples allowed in a relation.

The language used in INGRES is called QUEL. There are 2 types of commands in QUEL:

1. Those which create new relations with the create and retrieve commands.

2. Those which update existing relations with the <u>append</u>, <u>replace</u> or <u>delete</u> commands.

The specification of QUEL will not be presented here. Instead, this section offers some examples to give a flavor of the language.

An important part of most QUEL commands is called the <u>qualification</u>. The qualification is a condition that specifies exactly which tuples are used in a command. In a retrieval, the qualification defines which tuples are retrieved. In a deletion, it specifies which tuples are to be deleted. The qualification is a crucial component of QUEL. It is also the most complex component.

A second component is the target. The target specifies how to store retrieved data. For updates, the target specifies how to update the attributes. Some examples of QUEL statements will follow using the relations:

```
EMP(NAME,SAL,BDATE,MGR)
NEWEMP(NAME,AGE,SALARY)
```

QUEL uses variables called range tuple variables. Such variables may take on tuples as values. The declaration of these variables restricts them to a particular relation. The following range tuple variables will be used in the examples:

```
RANGE OF E IS EMP

RANGE OF N IS NEWEMP
```

The use of E in a QUEL statement implies that a tuple from EMP is to be used. The variable N implies a NEWEMP tuple.

Retrieve new employees under 25 and put them into a relation called YOUNG.

```
RETRIEVE INTO YOUNG (NAME=N.NAME,AGE=N.AGE)
    WHERE N.AGE < 25
```

All tuples of NEWEMP which pass the qualification (i.e. age is less than 25) are selected. From the selected tuples, a new relation called YOUNG is created. Each tuple in YOUNG has only 2 attributes:

```
YOUNG(NAME,AGE)
```

An example for updating is:

Give all employees who work for Jones a 10% raise.

```
REPLACE E (SAL=1.1*E.SAL) WHERE MGR="JONES"
```

All tuples of EMP whose manager is JONES are selected. The attribute SAL of each of these tuples has its value replaced by a value that is 10% higher than it was previously. The effect is a 10% raise for employees of JONES.

Qualifications and targets may become very complex, involving several relations at once. The following example uses both EMP and NEWEMP.

```
     Which managers have new employees under 25?

     RETRIEVE INTO X (MANAGER=E.MGR)

        WHERE N.AGE<25 AND N.NAME=E.NAME
```

This retrieval finds all tuples of NEWEMP which have AGE less than 25. It then finds all tuples of EMP which have the same name as the tuples selected from NEWEMP. The managers of these employees are collected into the new relation X. X has only one attribute named MANAGER.

The power of QUEL goes far beyond these examples. Readers who are interested in a specification of the QUEL language are referred to [HELD STONEBRAKER WONG] which contains a description of QUEL and further references.

5.2 Basis of the dual representation of data

The SDMS allows the tuples of a relation in the DBMS to be represented in the graphical data space. The graphic representation of each of these tuples is an icon. Those tuples having a graphic representation are distinguished from other tuples and are called <u>entities</u>. The icons that represent entities are distinguished from other icons and are called <u>entity-icons</u>. Each entity may be represented by more than one entity-icon. Conversely, each entity-icon may be represented by more than one entity.

Entities are simply tuples that have a corresponding icon. They may be used to represent real world objects, relationships between objects or anything else that may be represented in a tuple.

A goal of SDMS is to allow a user to refer to shared data via graphic methods, such as pointing, or via symbolic methods, such as using a QUEL retrieval statement. To achieve this, SDMS provides a mapping between each entity and its entity-icon. This mapping is provided via a <u>link</u> which logically connects the two. When an entity and entity-icon are linked, a selection of one implies a selection of the other. Entities are selected via an INGRES query. Entity-icons are selected via some graphical indication. In the query language of SDMS, the two selection methods may be combined through the use of links.

Links are created by <u>associating</u> a tuple with an icon, which makes the tuple an entity and the icon an entity-icon. There are two types of associations in SDMS. The first is a <u>specific</u> <u>association</u> which links one tuple to one icon. To perform this association, the tuple must already exist in some relation and the icon must already exist in some I-Space. The specific association is explained in Section 6.1.

The second type, called a class association, associates a relation with an I-Space. The relation and I-Space must already exist. When this type of association is made, SDMS creates a new icon for each tuple in the given relation. Each of these icons is placed in the given I-Space and linked to its corresponding tuple. Furthermore, the class association remains in effect until it is explicitly broken. If the given relation has tuples added, new icons are created and linked to the new tuples. If tuples are deleted, the corresponding icons are erased from the I-Space. Modification of a tuple linked to an icon causes the appearance of that icon to change.

Specific associations are most useful for entering symbolic data for use in locating graphical information, as in entering a description of the subject matter in a photograph. Class associations provide a means of creating graphical data for use in finding symbolic data. They eliminate the need for explicitly creating and linking an icon to each tuple by providing an automated mechanism for doing so.

The appearance of the entity-icons which are created by a class association may be controlled by the use of icon classes. An icon class is an abstract description of an icon, which contains instructions describing how each icon should appear. Icons which are created from an icon class may all look alike, or they may all be different. In the latter case, the appearance is dependent upon the data found in the corresponding entity. However, it is not necessary to specify how the entity-icons should appear. SDMS will provide a default appearance. The specific details will be determined as the system is built and utilized. One such default might be to use a rectangle with the values of the fields of the entity printed inside the rectangle. A full discussion of icon classes is presented in Section 7.

5.3 Commands and Languages

The primary usage of SDMS will be through its query language, called SQUEL. Its name arises from Spatial QUEry Language, or Spatial QUEL. SQUEL combines a symbolic query language and graphical interactions resulting in a uniquely powerful query language. Section 8 gives a full description of SQUEL.

A portion of languages of SDMS are dedicated to creating and editing I-Spaces, associating relational data with spatial data and describing icon classes. These generally fall into the realm of the <u>database administrator</u> although the capabilities may be delegated to the owners of individual I-spaces. The creating and editing of I-Spaces is described in Section 9.3. The commands for associating spatial and relational data are discussed in Section 9.2. The icon class description language is presented in Section 7.2.

6. ASSOCIATING SPATIAL DATA WITH SYMBOLIC DATA

The concept of associating data in the DBMS with data in the GDS was introduced in Section 5.2. This section will explain the process of association in detail. The syntax of the statements for performing associations can be found in Section 9.2.

6.1 Specific Associations

A _specific_ _association_ between a tuple and an icon creates a _link_ between the two. The tuple must already exist in some relation at the time of the association. The icon must also exist in some I-Space at the time of the association.

An important characteristic of a specific association is that the system does not maintain any data dependence which may exist between the entity and the entity-icon. This is quite different than the class association (Section 6.2). The specific association is primarily intended for use when the user is explicitly providing the entity-icon, rather than having the system generate it. For example, photographs in the GDS may be catalogued in a particular relation according to the subject of the photo. Each photo can have a corresponding entity in that relation so that a photo selection can be done via a symbolic query. This would be set up by establishing a specific association between between each photo and the tuple which represents it.

A second important characteristic has to do with the effect of changing either the entity or its entity-icon. Since there is no data dependence between them, updating one will not affect the other. Deleting one of them will <u>not</u> delete the other. Instead, the <u>link</u> is deleted and the specific association is "forgotten". Again, this is quite different with the class association.

6.2 Class Associations

The class association is an important concept in SDMS. It offers the principle tool for connecting the GDS to the DBMS. It is intended primarily as a tool for the <u>database administrator</u> (DBA).

A <u>class association</u> between a relation and an I-Space causes the creation of icons which graphically represent the tuples of the relation. The icons are placed in the given I-Space. The relation and I-Space must already exist at the time of the class association. It is possible to associate only a subset of the relation by supplying a qualification which determines which tuples are to be represented. It is also possible to have the placement of the new icons restricted to a particular rectangular region of the I-Space. The effects of such an association are:

1. For each tuple in the relation, an icon is created and inserted into the I-Space. If a qualification was supplied, icons are created for only those tuples which pass the qualification.

2. Each of these tuples and their corresponding icons are linked, so the tuples become entities and the corresponding icons become entity-icons. The system will maintain the correspondence between the two as the database is updated.

6.2.1 Usage of icon classes

When a class association is made, an icon class may be specified. The icon class describes exactly how to draw each icon. If an icon class is not specified, SDMS will use a default appearance for the icons. The details for the default appearance will be determined after some experience has been gained with using the system.

The icon class is essentially a description of a picture where certain parameters of the picture may vary each time the picture is drawn. The parameters include size, color and orientation. The values for these parameters may be the same for all icons which are created from the icon class, or the values may depend on the data in the entity being represented. Icon classes are discussed in Section 7.

6.2.2 Dependency between entities and entity-icons

When a class association is made, an explicit data dependence is established between each entity and its corresponding entity-icon. This is manifested at first when the system draws these entity-icons such that they reflect the data in the entity. Secondly, the data dependence serves to insure that the entity-icons which are created due to a class association will <u>always</u> reflect the tuples they represent, *until the association* is explicitly broken. Hence, if an entity is updated, its corresponding entity-icon is updated. If an entity is deleted, its entity-icon is erased. If new tuples are added to the relation, they become entities if they pass the qualification and new icons are created to represent them. The result is that the given I-Space always contains a representation for the given relation. Thus the I-Space serves as an alternate way to view the relation.

6.2.3 Flexibility of the appearance of entity-icons

The parameters of an icon class are listed in Section 7.1.2. If the user, particularly the DBA, wishes to change one of these entity-icons after it has been created, he may do so under certain circumstances. Since these entity-icons have characteristics that depend upon data in the corresponding entity, changing the appearance could be used to update the entity. The initial implementation of SDMS will employ a simple protocol for such updating. The user will be able to point to the part of an icon which contains the value of a field, in order to indicate which field to update.

Other changes may be made to the appearance of entity-icons for the purpose of making the I-Space easier to use. For example, the DBA may wish to change the color of a particular entity-icon without affecting the other entity-icons from the same icon class. Such a change can be made with the following restriction. The components of an entity-icon which are parameters of its icon class and which are dependent upon data may not be altered. This restriction ensures that the entity-icon truthfully represents its corresponding entity. This restriction would prevent the DBA from changing the color only if the corresponding icon-class used data to determine that color. If the color was not dependent on data (i.e. the color was constant), then he is free to change the color. Note that the position of each entity-icon is an exception to this rule and may always be changed.

7. ICON CLASSES

An icon class may be used in conjunction with a class association, as described in Sections 5 and 6. The icon class contains instructions which describe how an icon is to be drawn. While specific and class associations are relatively simple and may be performed by ordinary SDMS users, icon classes are more complex and are meant to be utilized primarily by the database administrator. The manner of describing an icon class to SDMS is the major topic of this section.

Essentially, a description of an icon class consists of several pictures, where certain parameters of each picture, like color or size, may be dependent upon the data that the icon represents. The icon class description includes rules which, for each tuple in a relation to be represented, decide which picture to use, which colors to use, how large to draw the picture, etc.

Take the following example. We have a relation containing data about ships -- destroyers and carriers. These ships are from several countries. An icon class description for this relation could specify the following:

- one picture is used for destroyers and another one for carriers
- the color of a flag on top of each picture is different for each nationality
- the size of each carrier grows proportionally to the number of planes on it
- the size of each destroyer grows proportionally to the number of men on it.

An icon class could easily be defined to accommodate these requirements. This example is used in Section 7.1.2.

When an icon class is declared, it is done with respect to a particular relation and I-Space. This is necessary because the icon class will normally depend heavily upon them.

7.1 The components of an icon class description

The primary components of an icon class are the pictures to be used. The requirements for pictures plus the ability to choose among them is explained in Section 7.1.1.

Each icon class has parameters which the user may control by providing rules that describe the values of the parameters for each icon drawn. These rules are expressions involving the data in the tuple being represented by the icon. In this way, the appearance of the icon can reflect the tuple as the user desires. The parameters are discussed in Section 7.1.2. These rules can also involve other data, not found directly in the tuple but elsewhere in the DBMS. This feature is described in Section 7.1.3.

Finally, icon classes may provide for drawing sub-icons. A sub-icon is an icon which overlays part of the original icon. This facility, discussed in Section 7.1.4, allows complex drawings to be constructed.

7.1.1 The pictures for an icon class

The pictures are the most important part of an icon class. Section 9 includes a description of the program for drawing pictures.

An icon which is constructed from an icon class has image data on each image plane in its I-Space. This allows the "zooming" effect by which a user sees a more detailed version of an icon by "flying" closer to it. The pictures for each image plane are defined as part of the icon class. For example, an icon for buildings might be drawn as follows: At the top-most (least detail) image plane, the building appears as a dot with its name below it. The second image plane has a rectangle with the building's name and address within the rectangle. The third (most detail) image plane is a more detailed picture including doors, windows and a smokestack. The building's name, address, size and type are written alongside the picture. With such a description, a user who "zooms" in on such an icon will get more detail as he gets closer. In summary, each <u>picture choice</u> consists of several drawings, one for each image plane in the I-Space.

Continuing with this example, we may want different drawings for different types of buildings. An icon class may have many picture choices, providing that a rule is included to select one for any given tuple to be represented. In such a case, the icon class description would contain several descriptions of picture choices, where each picture choice contained a drawing for each image plane.

7.1.2 Parameters of an icon class

The parameters of icon classes fall into two categories. As stated earlier, an entity-icon has a representation on every image plane of an I-Space. One category of parameters, called group parameters, affects the icon on every image plane of the I-Space. The other category, called picture parameters, affects the icon on one image plane only. Note that there are several references to the example presented in the introduction to Section 7.

The group parameters are:

1. _maximum size_: The maximum size of each icon may be specified to prevent any one of them from becoming too large. Since relative size is a picture parameter, it is a good protection. The default for the maximum size will be determined empirically.
2. _choice of picture_: There may be several different pictures to choose from depending upon the data in the tuple. In the example found in the introduction to Section 7, carriers had a different picture from destroyers. If more than one picture is given, a rule for deciding which picture to use must be provided.
3. _target position_: The target position is a rule specifying the exact location for an icon within its I-Space. SDMS does not let entity-icons overlap, so if an icon can fit at the target position, it will be placed there. Otherwise, a location close to the target position is used. The default is to place the icons in a standard place in the I-Space, possibly in a corner of the I-Space, and then to ask the user to assist in placing the icons.

The picture parameters are:

1. picture: This is the picture used to draw the icons. It is created by using the program for drawing (see Section 9.3).
2. color: Any region of the picture may either be the same color for all icons, or it may be colored differently depending on the tuple. In the example, the nationality determined the color of the flag, but all other parts of the picture were the same color for all icons. The default colors are the ones in the original picture.
3. size: The picture given for the description of an icon class may be scaled if the user wishes. He simply provides a rule which yields the desired scale. In the example, carriers with more planes were larger. The default size is the size of the original picture, up to the maximum size for the icon class.
4. orientation: The picture may be oriented by some arbitrary angle, if the user wishes. He simply provides a rule for the orientation angle. The default is the orientation of the original picture.

5. <u>text</u>: Text may be added to the picture. This text may be a simple string or it could be the data from one or more fields of the tuple.

6. <u>free text region</u>: During database queries, it is possible for retrievals to appear on the graphics screen next to the appropriate icons (see Section 8). The free text region is the area used for displaying this information. The user may have it wherever he wishes, as long as it is within the maximum size of the icon. This defaults to a region that, hopefully, does not overlap the icon. This depends upon how much room is available.

7. <u>attribute region</u>: It is used to display the value of one attribute of the entity tuple. It is useful when used in combination with the <u>change</u> statement of SQUEL. This allows the user to update the attribute by pointing at it and then entering the new value.

8. <u>draw statement</u>: calls a picture function and displays the picture generated by it.

7.1.3 Using INGRES data

One may make each entity-icon reflect the data in the entity it represents. This is done by making the data in the entity accessible from within the icon class description. It is also possible to retrieve other tuples from any relation in INGRES to aid in the description of an icon class. There are two statements to perform retrievals. One method is via the get statement, described in Section 7.2.6. The get statement retrieves one, and only one, tuple. If more than one tuple satisfies the qualification of the statement, an error occurs. Another method retrieves a set of tuples from a relation. They are accessed via the for loop statement. This statement has one or more statements included which are executed once for each tuple retrieved.

One must be cautious of the get and for statements because the icon class will be dependent on more than just the original entity. For example, an icon is linked to a tuple, but in the icon class description, another tuple (call it X) was retrieved and used to generate the icon. Now, X is updated and changed. The icon must now be updated to reflect the change in X. The result is that each entity-icon in this icon class is linked to an entity, but it is also dependent upon other data. The implications of these dependencies are discussed further in Section 7.3.

7.1.4 Sub-icons within an icon class

A feature of an icon class is the ability to have sub-icons drawn within the area of an icon. This feature allows the nesting of icons, for example displaying a sub-icon for each employee within a department that is represented by one icon. The user supplies a normal icon class description which will draw a picture on each image plane in the I-Space. Sub-icons may be included to overlay portions of the picture for some or all image planes. The "I-Space" for a sub-icon is the original icon, hence the sub-icon must be placed somewhere in the bounds of the original icon.

Each sub-icon is an entity icon. The statement used to declare sub-icons specifies the entity tuples for each sub-icon and the location for the drawings. The entity - sub-icon differs from a normal entity icon in that:

1. If the "parent" icon is moved, the sub-icons move with it.
2. If the "parent" icon is erased, and its link broken, all of its sub-icons are erased and their links are broken.

7.1.5 Picture Functions

For those instances where the parametric definition of pictures is appropriate, but no database dependencies are introduced, picture functions are provided. They are similar to sub-icon descriptions but receive all of their input values as parameters instead of INGRES tuples.

7.2 Icon Class Description Language

This section will give a detailed description of the language used to describe icon classes. This language should not be confused with the manner in which the description is entered. The entering of a description is done interactively, with much assistance given by SDMS. The final product of entering a description is an icon class description in Icon Class Description Language (ICDL).

Each icon class description will have a similar format. As an introduction, the typical format is now presented. The different parts of the description will be explained throughout this section.

```
icon class I(X) of relation R
   begin
      group parameters
      picture 1
         image plane 1
            graphical picture identifier
            picture parameters
         image plane 2
            graphical picture identifier
            picture parameters
         :
         :
      picture 2
         :
      :
   end
```

ICDL is a block-structured language. The icon class description forms the major block. Each picture description forms a picture block, and image plane descriptions form image plane blocks. The location of each block is crucial. For example, all image plane blocks for a picture must be contained within the corresponding picture block. The scope of tuple variables also depends upon the block structure, as is explained in Section 7.2.1.

The following sections will describe the language in an augmented BNF. Underlined words are terminal symbols and appear exactly as they do in the language. Other words are meta-linguistic symbols. The notation <stmt>* implies either one statement or a group of statements with begin and end around them, separated by semi-colons. Parentheses () appear in the language just as they do in the definitions. Square brackets [] imply optional components of statements. Curly brackets {} imply a choice among what they enclose. Comments may be included anywhere in the language. They are delimited by:

```
/* comment */
```

7.2.1 Tuple variables

Each icon which is drawn from an icon class description (call these generated icons) may reflect the data that it represents. To do this, the icon class description needs a method of accessing the data in the tuple it represents. In the icon class declaration, the notation I(X), where I is an icon class name and X is a tuple variable, specifies that during the generation of each icon, X is to be bound to the corresponding tuple. The effect is as if X was declared as a range variable for the relation with which the icon class is associated.

For example, let the relation R be defined as:

```
     R(F1,F2,...Fn)
```

In the icon class description, one may use:

```
     X.F1, X.F2,...X.Fn
```

as variables. They correspond to the attribute values of the tuple being represented.

This syntax also applies to tuples retrieved via the <u>get</u> statement and <u>for</u> statement.

Note that these values may used freely, but they may not be changed.

7.2.2 Icon Class Declaration Statement

```
    <icon class declaration> ::=
          icon class I(X) of R is <description block>
where
  -  I is the name of the icon class
  -  X is the tuple variable
  -  R is the name of the relation to be used with I

    <description block> ::=
          <global stmt>*

    <global stmt> ::=
          <maximum size stmt> ! <target position stmt> !
          <picture choice stmt> !
          <picture declaration stmt>!  <range stmt> !
          <get stmt> ! <for-1 stmt> ! <sub-icon stmt>
```

7.2.3 Global statements

These are the statements that are allowed in the first level of the icon class description. Some of them are the statements which control global parameters.

1. <u>maximum</u> <u>size</u>:

   The maximum size of the icon should be included as a protection against any one icon from taking up excessive parts of an I-Space. The maximum size defines a rectangle which the icon is restricted to.

   ```
       <maximum size stmt> ::=
           maximum size is (width,height)
   ```

   where width and height are arithmetic expressions.

2. <u>target</u> <u>position</u>:

   The target position statement is a rule which tells SDMS the exact location in the I-Space where to place the icon. If it would overlap another icon, a place near the target is chosen.

   ```
       <target position stmt> ::=
           position is (x,y)
   ```

   where x and y are arithmetic expressions.

3. picture choice:

Since an icon class description may have several pictures to choose from, a statement is included to decide which one to choose. It is necessary only if there is more than one picture in the icon class description.

```
<picture choice stmt> ::=
          use picture p
```

where p is an expression which results in a name of a picture. It is called a picture-name expression (see Section 7.2.8 for an explanation of expressions).

4. picture declaration:

This statement simply declares a new picture and allows it to have a name.

```
<picture declaration stmt> ::=
          picture name <picture block>
```

where name is the name of the picture. The name is optional.

```
<picture block> ::=
          <picture stmt>*
```

5. picture statements:

These are the picture statements. They are described later in this section.

```
<picture stmt> ::=
        <image plane description>  ! <range stmt>
        !    <get stmt>    !    <for-1 stmt>    !
        <sub-icon stmt>
```

7.2.4 Picture statements

The following statements are specific to a particular picture being included in an icon class. These include the statements for specifying picture parameters.

1. _image_ _plane_ _description_:

   For a picture choice, there is one drawing for each image plane in an I-Space. This is accomplished by requiring an image plane description for each image plane. Image planes that do not have such a description will not have any picture included.

   ```
       <image plane description> ::=
           image plane n <image plane block>
   ```

   where n is the number of the image plane in the I-Space. They are consecutively numbered starting from zero.

   ```
       <image plane block> ::=
           <image array> <image plane stmt>*
   ```

2. <u>image</u> <u>array</u>:

The image array contains the actual bit pattern which will be used to generate the picture. It is created using the programs described in Section 9.3. The appearance actual display of the image array is determined by the various image plane statements which follow it.

3. <u>image</u> <u>plane</u> <u>statements</u>:

These are the statements allowed for the description of one picture choice of an icon class on a particular image plane.

```
     <image plane stmt> ::=
          <color stmt>      !     <scale stmt>      !
          <orientation stmt> ! <text region stmt> !
          <display stmt>    !   <free text stmt>    !
          <range stmt>  ! <get stmt> ! <for-2 stmt>
          !   <sub-icon stmt>   !   <draw stmt>    !
          <attribute region stmt>
```

4. <u>color</u> <u>statement</u>:

Used to control the color of any region of the picture for this image plane.

<color stmt> ::=

<u>color</u> <u>of</u> region <u>is</u> C

where region refers to a region of the drawing and C is an expression whose value is a color. This statement will never be typed in explicitly but will be given through interacting with SDMS so the region and color may be specified by pointing to the graphics screen.

5. _scale_ _statement_:

   <scale stmt> ::=

   scale _is_ S

   where S is an expression giving the scale, or relative size, for the drawing.

6. _orientation_ _statement_:

   <orientation stmt> ::=

   orientation _is_ O

   where O is an expression that gives the orientation angle.

7. _text_ _region_ _statement_:

   Used to have text written over a portion of the picture.

   <text region stmt> ::=

   text _region_ [region-name] _from_ (X1,Y1) _to_ (X2,Y2)

   where a text region is a rectangle in which to place text. Text is placed in a text region by the _display_ statement. The text region is a free-format area for putting text. When something is added to it, the new text follows whatever has been written so far. The display statement has a few formatting controls.

8. display statement:

Used to display text or numbers in a text region.

```
<display stmt> ::=
        display S in T
```

where S is a string expression and T is a text region identifier. There are some pre-defined strings that are useful for formatting displayed information.

- NEXT -- forces the next string to be displayed to be put on the next line of the text region.
- TAB - has the effect of inserting a tab character.
- SPACE - inserts a space.

9. free text:

The free text region is a rectangle that SDMS uses for displaying the answers to queries. This region should not cover any important parts of the drawing since writing will occur in this area.

```
<free text stmt> ::=
        free text from (X1,Y1) to (X2,Y2)
```

Where these two points delineate the lower left and upper right corners of a rectangle which will be the free text region. These points are not typed in but they are pointed to by the user.

10. attribute region:

An attribute region is used to display one attribute for the entity tuple. Its main usage is in conjunction with the change statement of SQUEL (see Section 8.1).

attribute region Z.attribute from (X1,Y1) to (X2,Y2)

where Z.attribute specifies an attribute of the entity tuple.

11. draw statement:

Calls a picture function, generates a picture and places it onto the current image plane. Picture functions are described in Section 7.1.5.

draw P(arguments) from (X1,Y1) to (X2,Y2)

draws the picture generated by the picture function P in the region specified. The arguments to P cannot be tuples, only numbers or strings.

7.2.5 Range statement

The range statement is included in the ICDL so variables that access INGRES data may be declared. It is taken directly from QUEL.

```
     <range stmt> ::=
          range of X is R
```

where X is a tuple variable and R is a relation.

The scope of tuple variables in range statements is limited to the block in which they are declared.

7.2.6 Retrieving data from INGRES

The icon class description language offers two statements that allow INGRES data to be retrieved and used in an icon class description. The first is a get statement.

```
     <get stmt> ::=
          get Z where Q
```

where Z is a range variable that was previously declared and Q is the qualification which determines which tuple is retrieved. This statement retrieves *one and only one* tuple and binds it to the tuple variable. If no tuples pass the qualification or if more than one pass, then an error occurs. One may use this tuple variable in expressions as one uses the icon class tuple variable (see Section 7.2.1).

The *second* statement used for retrieving INGRES data is the for statement. There are two types of for statement. The for statement is actually a loop statement. It retrieves a set of tuples from INGRES and executes a group of statements, once for each tuple retrieved. The two types of for loop arise from the types of statements allowed in the body of the loop.

```
    <for-1 stmt> ::=

        for X [where Q] do

            { <sub-icon stmt> ! <for-1 stmt> }
```

where X is a range tuple variable that was declared previously and Q is a qualification. The effect of the for loop is that all tuples of the relation with which X is associated and which pass the qualification Q are retrieved. They form a temporary subset of the relation. This subset may be empty. The statements in the loop are executed with X bound to the first tuple in the subset, then re-executed with X bound to the second, etc.

The second types of for loop is:

```
    <for-2 stmt> ::=

        for X [where Q] do

            {    <display stmt>    !    <draw stmt>    !

            <for-2 stmt> }*
```

It behaves identically to the <for-1 stmt>.

7.2.7 Statement for using sub-icons

Sub-icons were introduced in Section 7.1.4. They allow an icon class description to invoke the same icon generation available to a user from SQUEL. The containing icon class description specifies which sub-icon class description to use, but the I-Space used is that of the current icon. Furthermore, only that portion of the I-Space is used that the current icon occupies. The sub-icon statement resembles the associate statement of SQUEL.

```
    <sub-icon stmt> ::=
        associate <tuple var> [where <qualification>
            [from (X1,Y1) to (X2,Y2)]
            [using <icon class>]
```

The <tuple var> must already be declared. The optional <qualification> allows the selection of which tuples are to be represented as sub-icons. The *from* clause specifies which portion of the invoking icon is to be used to display the sub-icon. If omitted, the system chooses a location. The *using* clause specifies which icon class description will be used to generate the sub-icon. If omitted, a default icon is used.

Using sub-icons is a powerful tool for drawing complicated pictures to convey information. In fact, the notion of sub-icons is recursive. Sub-icons may themselves have sub-icons. The resolution of sub-icons quickly goes to zero if one nests them too deeply, however. There is a price for using sub-icons. They introduce a new level of dependency that will be somewhat expensive to maintain. This is discussed in Section 7.3.

7.2.8 Expressions

Expressions in ICDL include the types of expressions found in most programming languages. They also have other operators that are useful for the purposes of the ICDL.

An expression is either a base value or an operator with one or more expressions. The number of expressions depends upon the operator. A base value is one of:

> integer, real (floating point), boolean, string, or picture name

The operators include:

> +, -, *, /, rem, ^, and, or, not, lt, le, eq, ge, gt, ne, ! (concatenation)

These operators also have the alternative representation of <, <=, >, >=, etc. The expressions which are used with these operators must be of the usual type. Other operators are:

<u>conditional</u> <u>operator</u>:

```
if C then A else B fi
```

where C is a boolean expression, A and B are the same (but any) type. The result is A if C is true; otherwise the result is B.

<u>case</u> <u>operator</u>:

```
case Y of
    X1 : V1
    X2 : V2
    :
    Xn : Vn
    default : Vd
esac
```

where Y, the Xi's and the Vi's are expressions. Y and the Xi's must be of the same type and the Vi's must be of the same type. This expression finds the first Xi that equals Y. The result is the corresponding Vi. If no Xi equals Y, then the result is Vd. Function calls may also be included in expressions. As yet, no facility has been designed for defining general functions but some system functions will be offered, for example trigonometric and logarithmic functions.

7.2.9 Labels

Any statement in the ICDL may have a label. The format is:

L : <stmt>

where L is a label. The major usage of labels is with the *like* macro. If you wish to use part of one statement in another statement, it can be done using *like*. For example, we have the statement:

L : *orientation* *is* E

where E is a complex expression. Now we wish to use the same expression elsewhere in the icon class description. This can be done by:

orientation *is* *like* L

which is equivalent to:

orientation *is* E

The *like* clause may be used with any statement in ICDL.

7.3 Dependency of an icon class

Each entity-icon that is generated must be truthfully represented in the GDS. The appearance of a entity-icon will be dependent upon data in the DBMS. If an icon class description performs retrievals from INGRES, this dependency can become quite complex.

The methodology adopted for maintaining the integrity of each entity-icon is to note which information it is dependent upon. Whenever that information is changed in any way, the icon is re-generated. In the future, strategies may be adopted which recognize when the changes to information will actually cause changes in the appearance of these icons.

This section will introduce an example which will be used throughout. Assume we have a personnel database with the following relations:

```
    PERSONNEL(name,dept,salary)
    DEPTS(deptname,manager,size)
    OFFICES(name,number)
```

which contains personnel names, salaries, managers and office numbers. We will examine the dependencies of using different statements in the icon class:

```
     icon class P(X) of PERSONNEL
```

where P is the icon class name, X its tuple variable.

The first level of dependency is obvious. If there are any changes to the relation PERSONNEL, the icons generated by the icon class P may need updating.

Other dependencies may be introduced if a get statement is used. For example, suppose we wanted to display the manager's name in the corner of the icon. To get the manager's name, we can use the employee's department name and go to the DEPTS relation.

```
     range of M is DEPTS  ;
     get M where (X.dept=M.deptname)
```

Now M is bound to the tuple of DEPTS for the employee's department, and M.manager is the manager's name. If we wanted to display the office number of the manager as well, there are two ways we can retrieve the number.

```
     range of Z is OFFICES ;
     get Z where (M.manager=Z.name)
```

Alternatively, we could use

```
     range of Y is DEPTS ;
     get Z where
             (X.dept=Y.deptname and Y.manager=Z.name)
```

In either case, Z is bound to the tuple of OFFICES for the manager of the employee being represented.

The important fact to note is that the dependency of this icon class is now extended to the three relations, PERSONNEL, DEPTS and OFFICES and that changes in either relation may affect the icons drawn via icon class P. The repercussion of this dependency is an extra burden upon SDMS to maintain the integrity of the graphical representation of this data.

8. SQUEL - THE QUERY LANGUAGE OF SDMS

The query language of SDMS is a combination of QUEL, the query language of INGRES, plus additions made for the graphical environment of SDMS. The name SQUEL arises from Spatial QUEry Language.

All methods of querying SDMS are included in SQUEL. This includes the capability for "flying" through the GDS in search of data. That portion of SQUEL is described in Section 3. Statements from QUEL can be entered directly to SQUEL, since QUEL is a proper subset. The following sections describe the additional features of SQUEL that may combine QUEL and graphical input/output.

The graphical representation of relational data permits retrieval of that data by means of such techniques as:

1. the system's graphical indication of icons

   - blinking icons
   - framing (drawing a box around) icons

- displaying answers to DBMS queries next to the icons involved in the query
- causing an icon to emit a characteristic sound

2. the user's graphical selection of icons

   - pointing to icons
   - using only those icons which are on the screen

A QUEL query has two parts which are extended in SDMS, a command and a qualification. The command instructs which actions to perform; the qualification determines which data in the DBMS is affected. The commands are extended to include:

- blink icons
- frame icons
- go to (rapid transit) the area of the GDS where certain icons may be found
- display a retrieval on the graphics screen next to the icons.

The qualifications are extended to include:

- those icons (and hence tuples) which are blinking
- those icons (and hence tuples) which are framed
- those icons (and hence tuples) which are pointed to

In summary, the query language is expanded to form a combination of symbolic and graphical input from the user. The responses by the system combine them as well, making a rich query language for the SDMS user. The syntax of SQUEL statements that are not found in QUEL are presented in the remainder of this section.

8.1 Additional query language commands

Several commands are added to the list of QUEL commands for SDMS.

1. blink <tuple var> where <qual>
   It finds all the tuples which satisfy <qual>. Any icons in the current I-Space which correspond to those tuples will blink. They continue to blink until a null blink command is entered or until the user leaves the current I-Space. A null blink command is one which has no arguments.

2. _frame_ <tuple var> _where_ <qual>

   Similar to _blink_ except that the appropriate icons are framed, not blinked. Framing an icon simply draws a rectangle around it. Framing is erased similarly to blinking.

3. _display_ (<target>) _where_ <qual>

   This is used to print results on the graphics screen next to the corresponding icons. Only those tuples which have corresponding icons have the values displayed. <target> is restricted to be a list of fields of a relation. The information printed by _display_ is not erased until a null _display_ command is entered or by leaving the current I-Space.

4. _find_ <tuple var> _where_ <qual>

   Used for a rapid transit to the location of the icon corresponding to the tuple retrieved. If more than one tuple is retrieved or if it is represented in more than one place in the GDS, the user is asked to decide upon which one to go to. Confirmation is required before the move actually occurs.

5. _point_

   Informs SDMS that the user will point to an icon. When he does, the data in the tuple corresponding to the selected icon is displayed on the graphics screen, or on the text display.

6. change

Informs SDMS that the user will point to a position within an icon. That position must be an attribute region for an icon (see Section 7.2.4). This signals that the user will update that attribute. The user then enters the new value for the attribute and the update is made immediately.

8.2 Additional relations to use in qualifications

Qualifications may be enhanced by the use of system defined relations. These relations are pseudo-relations. They do not actually exist in the INGRES database. However, one may treat them as normal relations in queries to SQUEL.

For these pseudo-relations, we must introduce the additional syntax requiring a relation name because of the definition of a relation. The tuples in a relation must be of the same format. However, the icons which are blinking, for example, may be from several different relations. Hence, we cannot have one relation corresponding to all the icons which are blinking. The same restriction applies for framing, etc. The following is a list of relation names and the tuples they represent.

1. <u>blinking-R</u>: The set of tuples which correspond to icons which are blinking, and which are members of relation R. This includes icons which are blinking but may be off the screen currently.
2. <u>framed-R</u>: The set of tuples which correspond to icons which are framed, and which are members of relation R. This includes icons which are framed but may currently be off the screen.
3. <u>onscreen-R</u>: This relation contains the tuples of relation R which are currently on the screen.
4. <u>point-R</u>: The user may select a set of icons by pointing to them. Those that represent tuples from relation R are put into the point-R relation.

9. COMMANDING SDMS

This section will present an overview of the commands and languages of SDMS.

9.1 Types of commands and languages in SDMS

The types of commands and languages of SDMS are:

- SQUEL - the query language
- statements to make associations
- creating and editing image spaces
- icon class description language

SQUEL was described in Section 8. Associations are described in Section 6, the syntax for these statements is in this section. The commands for creating and editing image spaces is included in this section. The icon class description language is presented in Section 7.2.

9.2 Statements to make associations

There are two types of associations, specific and class. The statement for a specific association between a tuple and an icon is:

link <tuple variable> *where* <qualification>

The user will be prompted to point to an icon. This creates a specific association between a tuple and an icon. The given qualification must be specific enough to retrieve one and only one tuple, otherwise the association is not made. Note that the qualification for this statement is "forgotten" after the *link* is made, so if the relation is later updated such that the qualification would not retrieve the same tuple, this has *no effect* upon the link. Once a link is created via a specific association, SDMS makes no further checks upon it. This point is made because this is completely different with the class association.

A specific association may be broken, and the link deleted, by the following statement:

break *link* *of* <tuple var> *where* <qualification>

If the tuple retrieved has no specific associations, or if no tuples pass the qualification, or if more than one tuple passes, an error message is given and the statement has no effect. If the tuple retrieved has more than one specific association, the user is prompted to select, by pointing, which icon to break the link with. Another version of this statement is:

    break _link_

After entering this statement, the user selects an icon and if it has a specific association, it is broken. Ambiguity and errors are handled in a fashion that is similar to the previous version of this statement.

There are two statements for a class association. One may associate an entire relation with a portion of an I-Space:

    associate <relation> _with_ <I-Space>
            [_from_ (x1,y1) _to_ (x2,y2)] [_using_ <icon class>]

The use of icon class is optional, as is the restriction to a particular region of the I-Space. If the I-Space is not given, the current I-Space is assumed.

If only part of a relation is to be associated:

    associate <tuple var> _where_ <qual> _with_ <I-Space>
            [_from_ (x1,y1) _to_ (x2,y2)] [_using_ <icon class>]

This restricts the class association to a subset of the relation.

One may break a class association which has the effect of deleting all links that were created plus erasing all entity-icons that were drawn because of the association. The statement to break a class association is:

break association of <relation> with <I-Space>

If there was more than one class association between the given relation and I-Space, the class associations are printed and the user asked to choose the one to break.

9.3 Creating and editing the GDS

This section describes the facilities provided to enable the user to create and modify the Graphical Data Space (GDS) of the SDMS.

9.3.1 User Environment

The interactive input facilities are available to the user for use in any I-Space which he has permission to modify. These facilities are invoked via a menu selection or typed command and operate in whatever I-Space the user is in. The user is at all times free to move about the SDMS, allowing him to perform operations which involve more than one I-Space, such as copying information from one I-Space to another.

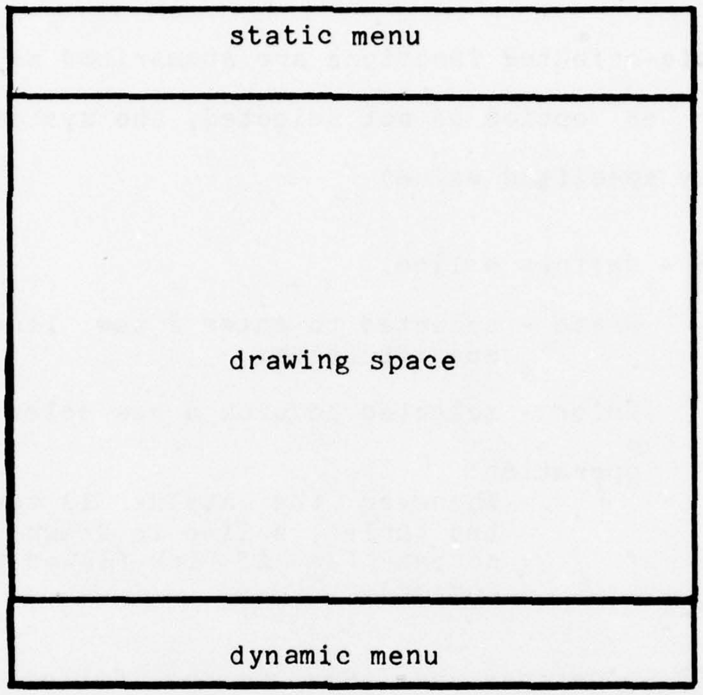
The actual writing of graphical data takes place in a "scratch" I-Space which functions in a manner similar to the buffer of a text editor, i.e. the database is not really modified until the buffer is explicitly saved, allowing the user to recover from errors and avoiding the possibility of two users modifying an I-Space at the same time. The usual mode of operation is thus to maneuver to an I-Space in the usual manner (using the joy sticks, traveling through ports, etc.) and issue the edit command. This command causes a menu of interactive input commands to appear on one of the auxiliary monitors, and it makes a copy of the I-Space currently on-screen. It is possible, however, to invoke the edit command in such a way that no copy is made -- so that the user can start with an empty buffer. This buffer can later be inserted into the SDMS database or used as a source of graphical data to be copied piecemeal into various I-Spaces.

9.3.2 User Interaction

Most communication with the interactive input facility is handled through the data tablet. The surface of the tablet is partitioned into three sections (see Figure 9.1).

The drawing space in the center of the tablet corresponds to the main monitor displaying the I-Space. Positioning the stylus in this area causes the cursor to move to the corresponding location on the screen. The permanent menu on the top contains commands which are always accessible from SDMS, such as returning to Unix, asking for help, and invoking the interactive input facility. The dynamic menu area is used to pick commands from changing menus on the auxiliary monitor.

tablet surface Figure 9.1



9.3.3 Description of Image Space Editing

The user is provided with a simple set of tools which can be used for generating and modifying all aspects of I-Space. These tools allow a user to work with general image areas when editing or with specific objects. The available menu functions can be split into two classes. Functions that are graphics oriented are designed for image generation. The other class of functions allow the creation of icons, i-planes and I-Spaces.

The graphic-oriented functions are summarized as follows: (whenever an option is not selected, the system uses the previously specified value)

LINE - defines a line.

Width - selected to enter a new line width specification.

Color - selected to pick a new color.

operation:
Whenever the stylus is touched to the tablet, a line is drawn on the screen (as if ink flowed from the cursor).

FLOOD - defines an area to be filled with a specified color.

Color - optionally selected to pick a new color.

operation:
The stylus is positioned within the area to be flooded and touched to the tablet.

PICK - defines an area or object to be copied

operation:
Three modes of the PICK command are available, to copy an arbitrary rectangular image area, to copy an icon, and to copy a plane. Two points must be entered for defining a rectangular area; otherwise, one will define the area to be copied.

PUT - defines an area or location for an object to be placed.

operation:
Three modes of the PUT command are available. To place an object with implicit scaling, the user enters an arbitrary rectangular image area into which the object is scaled. To just place an object at the same scale, only the center point is entered for the object. To place an object as a plane, one point anywhere on the drawing space is entered, the object is scaled to fill the screen (if smaller); otherwise, the scale is maintained.

TEXT - allows the user to insert text onto the scratch pad.

operation:
The user enters text on the keyboard while the cursor is positioned over the starting point for the text.

GRID - places a grid on the screen for use in alignment of objects.

Width - selected to define a new size grid to be used. The user enters two points defining the grid scale.

operation:
A Grid appears on the image plane (but is not stored in the graphical dataspace). When placing or drawing objects, positioning the stylus near a grid boundary causes the stylus location to "snap" to the boundary for easy alignment of objects on the i-plane and drawing of horizontal and vertical lines.

The remaining functions define the characteristics of a graphical data space.

MAKE ICON - defines a particular icon.

operation:
The area to be defined is pointed at with the stylus and the object becomes known to the system as an icon, which can be manipulated as one piece.

MAKE i-plane - defines a particular i-plane.

operation:
The user zooms the scratch pad I-Space to the desired detail for the plane and the menu command is entered. The user may choose to copy the data from the old i-plane to the new one. If the user is not on the scratch pad, a message is displayed and the command ignored.

MAKE PORT - defines a location through which a user may pass to enter a new I-Space.

operation:
Two independent operations are required: the first is to define the port origin; the second is to define the port destination. The port origin is defined by entering a window on an arbitrary i-plane, usually coinciding with an icon's boundary. The port destination is assumed to reside on the scratch pad. Once the port origin is defined, the user may move to the scratch pad where he may scale and position it to the desired view upon entering. A special function button defines the destination to be a Unix virtual terminal.

DELETE - allows a user to delete image areas, icons, i-planes and complete I-Spaces.

operation:

The user selects the type of deletion desired. For all except the image-area mode, entering the stylus position once on the given object will delete (and erase) it. If it's an image area, then 2 points are necessary for definition. For deleting I-Spaces and i-planes, confirmation is required.

9.4 Entering and editing icon class descriptions

The use of icon classes is encouraged because it is the primary tool for graphically representing relational data. The icon class description language may appear to be complicated but the important components are simple. The essential portions of an icon class description are the rules for determining the values of the parameters of an icon class. In an effort to make icon classes easy to use, SDMS will assist in the entering, and editing, of each icon class description. The mode of operation when entering a description will be highly interactive, where the interaction will remove many of the complications of the ICDL. In fact, it will be rare for the user to actually enter any ICDL statements. Instead, SDMS will prompt the user for the essential information and will automatically generate the statements. The example in Section 2.3 gives the flavor of this interaction by giving a step-by-step explanation of how a particular icon class description is entered.

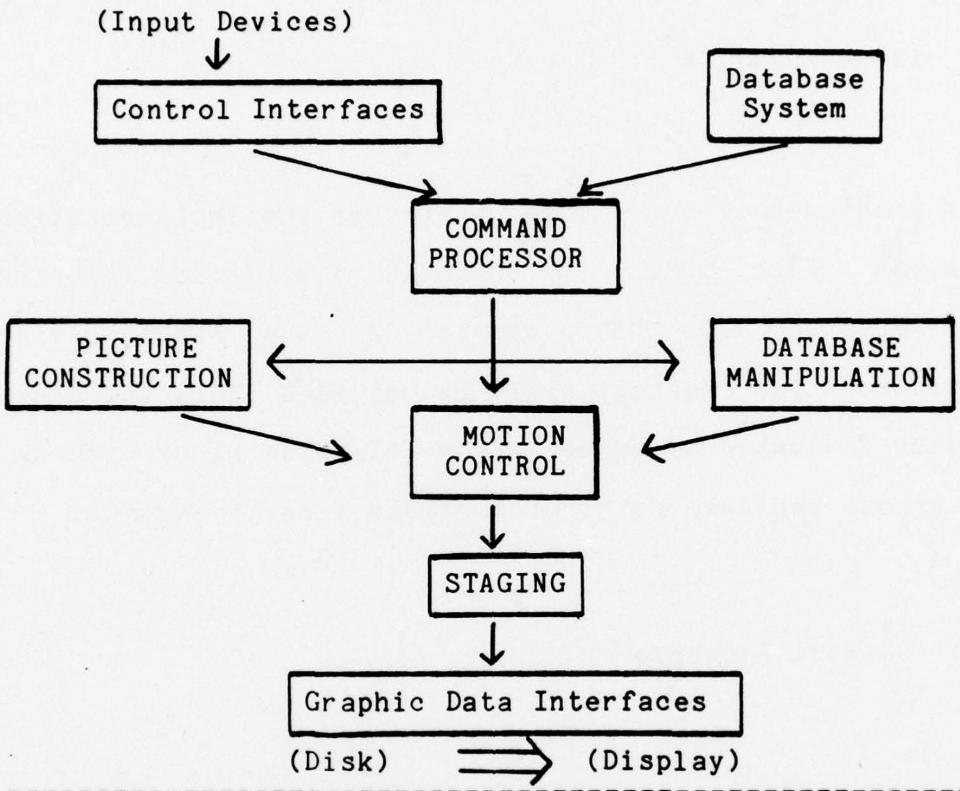
10. IMPLEMENTATION

This section provides a brief sketch of the implementation strategy for SDMS. A detailed description of the implementation will be presented in the Final Design Document. The system will be divided into the gross modules indicated by boxes in the following block diagram. The arrows indicate paths of possible flow of control.

10.1 Command Processor

The command processor (CP) is the interface through which all interactions with SDMS take place. There are two broad categories of such usage: direct manipulation by an SDMS user, and database updates generated outside SDMS. The CP functions primarily as a dispatcher to lower-level modules: Motion Control (including Icon Management and Navigation); Database Manipulation; and Picture Construction (including both direct user definition of icons, and automatic generation of predefined icons as the database is updated). It also serves to maintain the bindings of control device inputs (joysticks, tablet, etc.) to modules requiring inputs.

SDMS Block Diagram Figure 10.1



10.2 Control Interfaces

The devices available to the user to express his wishes will include joysticks, digitizing tablet, and function buttons, as well as a standard keyboard. Each of these devices requires its particular handler; in addition, SDMS will regularize the inputs from these devices as much as possible, so that the various modules which require information from them (e.g. motion control, icon definition) can accept standard inputs, and maintain some degree of independence from the particular devices.

10.3 Motion Controller

The motion controller consists of two main parts, which are closely coupled: the navigator, which is responsible for maintaining and manipulating the status information referring to locations in I-Space; and the icon manager, which, using locations provided by the navigator, maintains information regarding the icons which populate a particular I-Space. These functions are discussed in detail below.

The icon manager has access to a database which contains the location and description of each icon in the database. It can map in either direction between I-Space coordinates and icon-identifications. In the first capacity, it provides element identification to the query system when a query refers to an icon on the screen. Similarly, it informs the navigator of ports which are in the immediate neighborhood, so that transit through them may be properly set up. Since it has access to the definitions of icons, it can also provide facilities to the query manager for giving graphic responses to queries, e.g. changing an icon's color, or placing a text next to it on the screen.

The icon manager is also responsible for maintaining the icon database. In this capacity, it accepts instructions from the icon construction module to establish a new icon and find space for it. In a similar fashion, it will remove icons as data is changed or deleted in the underlying relation.

As the icon manager is to icons, so the navigator is to I-Spaces. It maintains a directory of the i-planes in each I-Space, as well as the neighboring I-Spaces which can be accessed through any port in the current I-Space. It accepts motion controls commands from the Command Processor, and passes the relevant motion commands to the stager; it maintains a clock so that the apparent motion is kept smooth and continuous.

The navigator is also responsible for displaying and updating the navigational aids provided with a given I-Space; this primarily involves reflecting the user's motion in the display of the current aid.

10.4 Stager

The stager oversees the actual transfer of data from the disk to the display, with any intervening modifications. In this capacity, it maintains a map between universal (I-Space) and screen coordinates. When requested by the motion controller, it instructs the display to scroll in either direction; to change the dimensions of the current viewscreen, and to zoom the picture in or out. If necessary for the requested motion, new information is fed to the display from the margin maintained in main memory by the stager; if that motion decreases the margin below a threshold, new information is read in from the disk in turn. If modifications are to be made to the image on the screen, these are passed to the stager in terms of either universal or screen coordinates, and it in turn triggers the appropriate update to the display, after updating its internal display buffer.

When the requested motion is a zoom, the stager consults a communication area maintained by the navigator to know whether and what view to prepare in the growing margin around the zoomed view.

The actual movement of data from disk to the display is carried on in an independent process, initiated and synchronized by the stager. The need for a disk transfer is recognized by the stager when its buffer margin decreases below a threshold, or when a zoom requires preparation of a new i-plane. A buffer for the transfer is selected from a pool and the disk data is read into it. The data will generally be encoded (three obvious encodings are run-length, fill, and identity); the code identification contained with the data will serve as a selector for an appropriate decoding function which will fill a portion of the display buffer maintained in main memory. A communication area maintained by the icon manager will be interrogated, and any icon-modification functions provided there will be applied. Eventually, some requested motion will cause part of the new image to be displayed; the stager will reformat it appropriately and start a transfer to the display device.

10.5 Database Manipulation

Data manipulation is the domain of the query manager. Database queries received by the command processor are given to it for evaluation and response. The actual generation of a query may involve both graphical and discursive elements. In resolving these into a simple query intelligible to the database management system, the query system may require information from the icon manager as to the identity and nature of indicated picture elements. It will prepare and submit requests to the DBMS in standard form, and receive and interpret the results. It may, again, consult with the icon manager if the response is to be given a graphic form. This may eventually provoke update requests to the stager which result in a change to the displayed scene.

A major area of interaction between the DBMS and SDMS is left outside the scope of the query system: creation and update of icons according to changes in the symbolic database will proceed as requests from the database system to the command processor, which will be handled primarily by the icon generation facilities of the picture construction module.

10.6 Picture Construction

Picture construction has two major phases, referring to the time an icon is defined, and the time it is actually generated and stored in the graphics database.

Icon definition occurs when a class association is created between a relation and an I-Space. The facilities provided for this manipulation will include the capability to move around existing I-Spaces, create new ones, create and remove i-planes in an I-Space, and build the archetype of a relation's icons in those i-planes. These manipulations will take place in a protected version of the I-Space; changes made by the user will not be visible to any other processes until the modifications are completed successfully. At that point, the new icon definition(s) will be added to the icon dictionary, linked to their relation(s) in the database, and the I-Space which has been used for their creation will be spliced or merged into the rest of the universe as appropriate. The language available for the definition of icons combines graphic and algebraic elements in a rich and powerful tool; it has been described in Section 4 of this paper.

Icon generation occurs as data is added to the symbolic database. It proceeds in the background, without requiring any user's attention, or even notice. When data is added to the database, from whatever source, it triggers a request to the command processor to update the graphics database. This turns into a request to the motion controller to generate an instantiation of the icon for the new tuple's relation, with location and variable aspects appropriate to the particular tuple. This in turn affects the icon manager and the navigator; the first must access the appropriate icon definition and pass to the navigator a description of the I-Space volume that will be affected. The navigator then gets access to the relevant spaces through the stager, and the icon manager proceeds to generate an icon (or a series of them) in it, possibly performing database queries along the way to determine particular aspects of the icon as it is formed. As each icon is completed, it is released to the stager, for update of the graphic database. Note that this automatically provides the user with new information as it becomes available. The same procedure can be invoked explicitly by a user wishing to hand-add an icon to an

I-Space, but this is not expected to be a common practice.

A. Generation of the Graphical Data Space

This section illustrates how icons can be created through icon class description. There are commands for creating new I-Spaces with a specific number of image planes, and for creating ports which connect the I-Spaces. Icons are entered either by drawing them with the aid of a program for drawing or by entering photographs.

In the example, the icons of I-Spaces which appear in Figures 2.3 through 2.10 were each entered individually in this fashion. The ports connecting these I-Spaces were also entered individually. The I-Space which appears in Figures 2.11 through 2.15 was partially created automatically through the use of a class association. The creation of this I-Space will be described in detail to give an idea of the usage of class associations and icon classes.

This I-Space contains a graphic representation of the PERSONNEL relation. It groups personnel by the project each person is associated with. There are 4 image planes in this I-Space. For reference purposes, the I-Space will be called "I-PROJECT", since it represents the projects of the SRD division. The image planes are numbered 0 through 3, which goes from least detail to most detail.

The icons on image plane 0 (Figure 2.11) were entered individually using the drawing program. This image plane divides the SRD division into projects. In the upper left corner is an icon representing the manager of the division. The other icons represent different projects. Most of the icons on image plane 1 (Figures 2.12 and 2.13), image plane 2 (Figure 2.14) and image plane 3 (Figure 2.15) were generated by SDMS through the use of a class association. The parts that were hand drawn are the boxes surrounding all personnel in each project and the text which gives the project name and project related information. The icons generated automatically are those which correspond to personnel. There was one class association made for each project and manager. The following are the range variables that are needed for each class association:

```
range of P is PERSONNEL
range of D is DIVISIONS
range of PR is PROJECTS
```

The following statements are needed to perform some of the class associations necessary for the SRD division, presented in Figure 2.11. The first statement is used for the manager of the SRD division.

```
    associate P
        where (D.division="SRD" and D.manager=P.name)
        from (X1,Y1) to (X2,Y2)
        using IC-PERSONNEL
```

The coordinates (X1,Y1) and (X2,Y2) delimit diagonally opposite corners of the icon for the division manager. These coordinates are not actually typed in but are entered to SDMS by pointing at the corners. The effect of this class association is to find all managers of the SRD division (there is only one) and to create icons for each according to the icon class IC-PERSONNEL. These icons are linked to the manager tuples and are placed in the specified region. We have specified this region to be under the icon for division manager. The description of IC-PERSONNEL is presented later in this section. It draws icons on image planes 1, 2 and 3 only, so it does not affect the icons on image plane 0.

The statement for making the class association for the SDMS project is:

```
    associate P where (P.project="SDMS")
        from (X1,Y1) to (X2,Y2)
        using IC-PERSONNEL
```

Again, the region for placing the new icons is specified by pointing, not by entering coordinates. Now the SDMS project will have entity-icons representing them under the

SDMS project icon. The other projects are specified similarly.

The icon class IC-PERSONNEL is relatively complex. Icons which were drawn from it appear in figures 2.12 through 2.15. It contains a photograph of each employee plus a collection of information. There are three regions to this icon class:

1. A text region appears on the top containing name and phone extension.
2. Photograph of employee.
3. A text region containing computer address, home address and computer background.

This icon class is defined for 3 image planes in the I-Space, image planes 1, 2 and 3. It does not affect any other image planes. The icon class must define these three regions and must describe what to place in each one.

The description of the IC-PERSONNEL icon class will now be presented. The presentation will simulate the manner by which it would be entered. There is a description of the icon class description language (ICDL) in Section 7. The language is not used directly for entering the description. Instead, we take advantage of the interactive graphical environment to ease this task. For the rest of this section, indented text is meant to be text that is either entered by the user or printed by SDMS. The text typed by the user is underlined. Some of the statements and commands need not be typed in by the user. At times, there will be a menu of commands on one of the graphics screens which can be selected by pointing at the command. Those commands that the user selects which appear on a menu will not only be underlined but will have curly brackets {} around them. An explanation of the process of entering the description will occur throughout the section.

Initially, the user is "talking" to the top-level in SDMS (explained in Section 9).

icon class

What is the name of icon class? IC-PERSONNEL

IC-PERSONNEL is a new icon class, please confirm: Yes

What is the name of relation to use with IC-PERSONNEL? PERSONNEL

What is the name of the PERSONNEL tuple variable? P

Which image space will IC-PERSONNEL be used with? I-PROJECT

At this point, the user will begin entering the icon class description. A menu appears on one of the auxiliary screens. It contains commands that correspond to group statements of the ICDL. On the other auxiliary screen appears the program for drawing. It is initially blank. The user enters the range variable declarations needed.

```
    range of PH is PHONE ;
    range of LN is LOGIN-NAME ;
    range of H  is HOME  ;
```

He then begins to enter the description of the picture.

```
    {picture}
        name of picture (optional) ?
```

SDMS has been instructed that a picture is about to be described. The user did not wish to give it a name. A name is not necessary because there will be only one type of picture in this icon class, so there need not be a rule specifying which picture to use. At this time, a new menu appears on an auxiliary graphic screen with the picture statements.

```
    {image plane}
        Enter number of image plane : 2
        Please begin drawing picture for image plane 2.
```

The user decides to first enter the description for image plane 2, leaving the descriptions of 1 and 3 for later. He is now encouraged to begin actually drawing the picture for image plane 2. A new menu appears with the image plane statements. First, he draws a rectangle which is the outline of the icon. Then he specifies how to draw the photo.

{sub-icon}

Enter name of icon class? PHOTO(P)

Do you wish to specify which region to place the PHOTO sub-icon? Yes

Please point to region.

The user points to the place where to place the photo. SDMS draws a lightly colored box around the region for the user's reference. The PHOTO icon class is assumed to have been previously defined for the PERSONNEL relation. It draws a photograph of the employee. He now defines the two text regions.

{text region}

Enter name of text region (optional) : TOP

Please point to the region for TOP.

The user specifies the region above the photo.

{text region}

Enter name of text region (optional) : BOTTOM

Please point to the region for BOTTOM.

The user specifies the region below the photo. After each text region statement, the system draws a lightly colored box over the text region. Now he specifies the text to place in each region.

{display} (P.name ! NEXT) in TOP

In this case, the user typed in the entire statement, so no interaction was needed from SDMS. The string expression is the persons name followed by an end-of-line marker. The "!" is the concatenation operator. The other information to be displayed must be retrieved from INGRES.

```
    for PH where (PH.name=P.name) do
        display (PH.phone ! NEXT) in TOP
    for LN where (LN.name=P.name) do
        display (LN.login-name ! " @ " LN.system)
        in BOTTOM
```

To display the home address and phone number, he uses a get statement.

```
    get H where (H.name=P.name)
    display (H.address ! NEXT ! H.city ! ", " ! H.state !
                SPACE ! H.zip ! NEXT ! H.phone-number)
                in BOTTOM
```

The description for this image plane is now complete. The user ends it by:

```
    {end} of description for image plane 2.
```

He follows this by the description for the next image plane.

```
    {image plane}
        Enter number of image plane : 3
        Please begin drawing picture for image plane 3.
```

The description for this image plane is similar to the one for image plane 2, except it has even more information displayed. The entering of this description is left out of this example since it is similar to the previous image plane. He follows this description by the description for image plane 1. This example continues after the description of image planes 1, 2 and 3.

{end} of description for picture.

{end} of description for icon class IC-PERSONNEL.

The icon class IC-PERSONNEL has been entered. Since no description was given for image plane 0, no picture will be drawn on that plane for this icon class. In this way, it will not affect the hand drawn icons on that plane. Note that it had to be defined before it was used in a class association. The IC-PERSONNEL icon class had no specification for target position. Since it was left out, SDMS selects the exact location of each icon created from it. This location will be within the region specified in the class association statement.

Figure A.1 shows the icon definition language statements which result from the preceding dialog.

ICDL for example of Chapter 2   Figure A.1

```
Icon class IC-PERSONNEL(P) of PERSONNEL

Range of PH is PHONE

Range of LN is LOGIN-NAME

Range of H is HOME

   Picture

      Image Plane 1
         Sub-icon PHOTO (P) from (X1,Y1) to (X2,Y2)
         Text region TOP from (X1,Y1) to (X2,Y2)
         Text region BOTTOM from (X1,Y1) to (X2,Y2)
         Display (P.name) in TOP
         For PH where (PH.name=P.name) do
            Display (PH.phone) in TOP
         For LN where (LN.name=P.name) do
            Display (LN.login-name "@" LN.system) in BOTTOM
         Get H where (H.name=P.name)
         Display (H.address H.city H.state H.zip
            H.phone-number) in BOTTOM

      End
      Image Plane 2
         .
         .
         .
         .
         .
      End

   End

End
```

B. GLOSSARY OF TERMS

class association. Associates a relation with an I-space by creating new icons for each tuple in the relation and placing them into the I-space. Furthermore, each icon is linked with its corresponding tuple in a way that the icon may graphically represent the tuple.

Database Administrator (DBA). Person responsible for the structure, content and use of a database installation.

DBA. Database Administrator.

Database Management System (DBMS). The part of an SDMS which stores all of the symbolic data. As distinguished from the Graphical Data Space (GDS) which contains all of the pictures, icons, ports, and text.

DBMS. Database Management System.

entity. A tuple of some relation which is linked to an icon.

entity-icon. An icon which is linked to a tuple of some relation.

frame buffer. A memory used for refreshing a raster-scan display, having one memory cell for each picture element (pixel). The color of each picture element (pixel) on the screen is determined by the value stored in its corresponding memory cell.

GDS. Graphical Data Space.

Graphical Data Space (GDS). The union of all of the I-spaces in an SDMS. The GDS contains all of the graphical information in an SDMS, as distinguished from the database management system (DBMS) which contains all of the symbolic data.

ICDL. Icon Class Definition Language.

Icon. A picture in an I-Space which indicates the location of information.

icon class. A icon type which is defined by the DBA. The definition includes a description of how to draw icons which are members of the class.

Icon Class Definition Language (ICDL). The mechanism by which icons can be created as a function of information in the symbolic database management system.

information-plane (i-plane). A bit-array which is a complete graphical representation of an I-Space at some particular level of detail. Multiple i-planes are presented on the user's display in a sequence which gives the appearance of increasing detail as the user moves closer to the apparent surface of the I-Space.

Information Space (I-Space). The basic unit of a Graphical Data Space (GDS) used for holding information. Each I-Space is a two-dimensional plane over which the user can move. I-Spaces are usually set up to contain pieces of related information.

INGRES. A relational database management system developed at the University of California at Berkeley.

i-plane. information-plane.

I-Space. Information Space.

navigational aid. A facility, using some device such as an auxiliary display or a sound system, which helps the user of an SDMS find his way around a GDS or a particular I-Space.

perusal space. A process which enables the user to interact with the display in a manner other than the usual I-Space manipulation. Some examples perusal spaces are a text editor, a message system, and an animated movie.

pixel. Picture element. The smallest area of a raster-scan display which can be independently changed. A standard television image has 307,200 pixels.

port. A location in an I-Space which is used to leave that I-Space for a new environment, typically another I-space.

QUEL. The query language of INGRES.

raster scan. The process of displaying an image by displaying successive lines which are scans of the image, each of which can vary in color and intensity along its length. As distinguished from a calligraphic display in which the electron beam traces out vectors which make up the image. Broadcast TV works by raster scan. Computer graphics has been dominated until recently by calligraphic displays such as storage tubes and so-called dynamic refresh displays.

relation. Intuitively, a table of rows and columns. The rows are called tuples and the columns are called attributes. Formally, given sets D_1, D_2, ... D_n (not necessarily distinct), a relation R is a set of n-tuples each of which has its first element from D_1, second element from D_2, etc. The sets D_i are called domains.

specific association. Links one tuple to one icon without introducing any data dependency of one to the other.

SQUEL. The query language of SDMS: a version of QUEL which has been augmented to include graphical capabilities.

tuple. An ordered list. For example:
(JONES,PROGRAMMING,ROOM 513)

References

[BOLT a]
Bolt, Richard A., Spatial Data-Management, Interim Report, MIT Architecture Machine Group, November, 1977.

[BOLT b]
Bolt, Richard A., "SDMS II", Internal Memorandum, MIT Architecture Machine Group, March 15, 1978.

[CCA]
"Design and Implementation of a Spatial Data Management System", proposal submitted to Cybernetics Technology Office, Defense Advanced Research Projects Agency, June, 1977.

[DONELSON]
Donelson, William C., "Spatial Management of Multimedia Information", to appear in proceedings of SIGGRAPH '78, August 1978.

[HELD STONEBRAKER WONG]
Held, G.D.; Stonebraker, M.R.; Wong, E., "INGRES - A relational data base system", *AFIPS Proceedings*, Volume 44.

[HEROT et al]
Herot, C.; Rothnie, J.; and Farrell, J., Quarterly Research and Development Report, Spatial Data Management System, Computer Corporation of America, February, 1978.

[NEGROPONTE & FIELDS]
Fields, Craig; and Negroponte, Nicholas, "Using New Clues to Find Data", Third International Conference of Very Large Databases, Tokyo, Japan, 1977.